NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

GENERATION OF DEPARTMENT OF DEFENSE ARCHITECTURE FRAMEWORK (DODAF) MODELS USING THE MONTEREY PHOENIX BEHAVIOR MODELING APPROACH

by

Joanne D. Pilcher

September 2015

Thesis Advisor: Kristin Giammarco
Second Reader: Walter E. Owen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2015	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
GENERATION OF DEPARTMENT OF DEFENSE ARCHITECTURE FRAMEWORK (DODAF) MODELS USING THE MONTEREY PHOENIX BEHAVIOR MODELING APPROACH

5. FUNDING NUMBERS

6. AUTHOR(S) Pilcher, Joanne D.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Center Pacific
53560 Hull Street
San Diego, CA 92152-5001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	**12b. DISTRIBUTION CODE**
Approved for public release; distribution is unlimited	A

13. ABSTRACT (maximum 200 words)

The Department of Defense (DOD) has struggled with the systems development, integration and interoperability for more than 30 years. Despite the Department of Defense Architecture Framework (DODAF) compliance requirement initiated 20 years ago to address these concerns, DOD agencies continue to struggle to deliver interoperable systems required for operations. The Monterey Phoenix (MP) approach shifts the paradigm underlying these DODAF views to focus on system behaviors and interactions rather than component functionality and the data flows between them. Although robust DODAF tools are available for model documentation, the MP Analyzer tool enables the system architect to reduce design complexity while quickly and easily exposing architectural flaws prior to implementation.

This research defines DODAF models that can be generated using the MP approach to realize MP benefits such as automatic scenario generation and comply with DOD guidance. Using criteria established in this research, 16 of the 51 total DODAF models from five of the eight viewpoints are produced using data available in the MP approach. The value proposition to DOD programs is the ability to intercept design errors before they become costly system failures or rework requirements. Future research can validate the DODAF model generation from MP as it matures.

14. SUBJECT TERMS	**15. NUMBER OF PAGES**
business process modeling notation, BPMN, Monterey Phoenix, department of defense architecture framework, DODAF, systems engineering, systems, systems architecting, architecture, validation, verification	131
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

A**pproved for public release; distribution is unlimited.**

GENERATION OF DEPARTMENT OF DEFENSE ARCHITECTURE FRAMEWORK (DODAF) MODELS USING THE MONTEREY PHOENIX BEHAVIOR MODELING APPROACH

Joanne D. Pilcher
Civilian, Space and Naval Warfare Systems Center Pacific
B.S., Virginia Tech, Blacksburg, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2015

Author: Joanne D. Pilcher

Approved by: Kristin Giammarco, Ph.D.
Thesis Advisor

Walter E. Owen, DPA
Second Reader

Ronald Giachetti, Ph.D.
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Department of Defense (DOD) has struggled with the systems development, integration and interoperability for more than 30 years. Despite the Department of Defense Architecture Framework (DODAF) compliance requirement initiated 20 years ago to address these concerns, DOD agencies continue to struggle to deliver interoperable systems required for operations. The Monterey Phoenix (MP) approach shifts the paradigm underlying these DODAF views to focus on system behaviors and interactions rather than component functionality and the data flows between them. Although robust DODAF tools are available for model documentation, the MP Analyzer tool enables the system architect to reduce design complexity while quickly and easily exposing architectural flaws prior to implementation.

This research defines DODAF models that can be generated using the MP approach to realize MP benefits such as automatic scenario generation and comply with DOD guidance. Using criteria established in this research, 16 of the 51 total DODAF models from five of the eight viewpoints are produced using data available in the MP approach. The value proposition to DOD programs is the ability to intercept design errors before they become costly system failures or rework requirements. Future research can validate the DODAF model generation from MP as it matures.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AV	all viewpoint
BPMN	business process modeling notation
CIO	Chief Information Officer
COTS	commercial off-the-shelf
CRADA	cooperative research and development agreement
CV	capability viewpoint
DIV	data and information viewpoint
DOD	Department of Defense
DM2	Department of Defense Architecture Framework Meta Model
DODAF	Department of Defense Architecture Framework
GOTS	government off-the-shelf
IDE	integrated development environment
IDEF0	integration definition for function modeling
INCOSE	International Council on Systems Engineering
LML	life cycle modeling language
OV	operational view
PES	physical exchange specification
PV	project viewpoint
SOA	service oriented architecture
StdV	standard viewpoint
SvcV	service viewpoint
SV	system viewpoint
MP	Monterey Phoenix
UML	unified modeling language

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The practice of systems architecture is in its infancy as discipline in the systems engineering community. Many examples of failed systems development efforts exist with large cost and schedule overruns. The Department of Defense (DOD) has been struggling with the development, integration and interoperability of its systems for more than 30 years. Twenty years ago, the DOD developed the Department of Defense Architecture Framework (DODAF) to help improve system development efforts.

Despite the DODAF compliance requirement and software tools to facilitate model development, DOD agencies continue to struggle in the delivery of quality and usable systems within the planned costs and schedule. Researchers at the Naval Postgraduate School (NPS) are using the Monterey Phoenix (MP) approach to shift the paradigm for architecture development and improve system development outcomes. At the core of MP is the principle of modeling behaviors and the interactions between them as the means to early discovery of architecture concerns. This research defines what DODAF models can be generated using the MP approach thus realizing the many MP benefits while meeting the DOD program compliance requirements. Additionally, the research develops the methods of conversion and provides recommendations for visualization usage and improvements in the MP prototype, the MP Analyzer.

As stated on the Monterey Phoenix website, MP provides an architecture approach with a focus on system behaviors and the interactions between these behaviors. System behaviors and interactions are modeled separately, allowing MP to automatically generate an exhaustive set of use cases at a small scope to identify system behaviors—a capability unique to the MP approach. This set of use cases provide the modeler the ability to visually determine the behaviors of the system either intended or unintended. In addition to the automatic generation of use cases, other advantages of the MP approach, as described on the MP website, include early assessment of non-functional requirements, early identification of design flaws with the potential to save costs on rework for errors found during implementation, support of reusable architectural patterns, ability to

integrate with standard notations such as UML and SysML, and the simplicity of the MP grammar.

DODAF consists of eight viewpoints and 51 related models. Criteria established in this research narrowed the list of models evaluated:

- the DODAF model is graphically represented;
- the model has implementation of precedence relations;
- the model has the implementation of inclusion relations.

Using these criteria, 16 of the 51 models were evaluated. A case study example of each model was developed, a method established for conversion of the case study example to MP, and MP code was developed and executed. Monterey Phoenix-generated visualizations were evaluated, and a summary of the results documented. Department of Defense Architecture Framework models can be generated from five of the eight viewpoints as shown in Table 1.

Table 1. Summary of DODAF models generated using MP.

Viewpoints	**DODAF Models Generated from MP**
All	None
Capability	CV-2
Data and Information	None
Operational	OV-2, OV-4, OV-5a, OV-5b, OV-6b, OV-6c
Project	PV-1
Services	SvcV-2, SvcV-4, SvcV-10b, SvcV-10c
Standards	None
Systems	SV-2, SV-4, SV-10b, SV-10c

While MP is able to generate all of the above models, the current MP Analyzer prototype visualizations are limited and primarily intended for academic use. Many other commercial tools present better graphical visualizations with more robust manipulation capabilities. Modelers may consider using alternative tools as the MP Analyzer prototype matures. Researchers hope to inspire systems architects, systems engineers, and industry to adopt the MP approach. With greater adoption of the MP approach, model based system engineering (MBSE) vendors can extend the MP capabilities and/or incorporate them in their own tools.

In conclusion, the complexity of today's systems development efforts demand better methods and approaches to simplify and improve successful outcomes. Significant improvements are simply not being realized. MP introduces a new approach to the mix and the results of this research reveal much promise for improving systems development through simple, early discovery of behaviors through the generation of an exhaustive set of use cases (trace events). The MP approach is still under development and planned extensions are already in development. As such, continued research to study and transform complex system architectures that are struggling to meet cost, schedule and performance requirements would be invaluable. Such a study will provide insight on the potential return on investment that can be realized using the MP approach.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I acknowledge and thank my family, my fellow cohort students, my co-workers, my employees, my dance moms, my dearest friends, and, last, but not least, the faculty at Naval Postgraduate School. Professors Kristin Giammarco and Mikail Auguston provided invaluable Monterey Phoenix guidance and code reviews for this research. I have learned much on this two-year journey. All of you have contributed directly to my ability to complete this master's program by providing me academic guidance, help, moral support and friendship, and sometimes, just a shoulder to lean on. I look forward to spending more time with my family and friends in the very near future and tracking the progression of the Monterey Phoenix approach.

I am most blessed and grateful that my father, who was diagnosed with lung cancer during the spring quarter of this year, is here to see me graduate. Finally, I thank the person who has shouldered the majority of the burden of my physical and mental absence during the past two years, and unselfishly picked up the slack, my dear husband and best friend, Daniel Pilcher.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

In the original predecessor to *The Art of Systems Architecting*, published in 1991, Eberhardt Rechtin opens with this statement: "Architecting, the planning and building of structures, is as old as human societies—and as modern as the exploration of the solar system" (Maier and Rechtin 2009, xv).

The building of structures includes the discipline of architecture and civil engineering and is a widely accepted practice. Building architects and civil engineers work hand in hand in the development of structures across the globe. Maier and Rechtin's (2009) opening statement suggested systems developers applied civil engineering architecting methods to systems development, inadvertently. Since the first release of their book, Maier and Rechtin's (2009) assumption regarding the application of civil engineering architecting methods to systems architecting continues to be validated through academic and industry studies. The complexity of systems development drives the need to apply architecture concepts to today's systems engineering approaches. Through the development of structured architecture frameworks and the application of heuristics, the value of systems architecture to the successful systems implementation continues to gain credible acceptance in the systems engineering community (Maier and Rechtin 2009).

Many organizations struggle to understand the distinction between systems architecting and systems engineering, thus the value of systems architecting is often questioned. The systems engineering team builds and delivers products. Their value is easily quantified. Architecture teams create documents, build models, consume resources, but their value is not always immediately tangible. To understand the partnership between system architecture and system engineering better, it is necessary to define their individual elements: system, system engineering, and system architecting.

1. Systems

Systems are everywhere in everyday existence. There are human systems, space systems, weather systems, software systems, hardware systems, solar systems—and the list goes on. However, defining "system" proves surprisingly challenging due to the almost infinite set of objects to which the concept can be applied. As such, many definitions with varying criteria exist in contemporary literature.

Blanchard and Fabrycky (2011, 3) define a system as "an assemblage or combination of functionally related parts forming a unitary whole, such as a river system, or a transportation system." They classify various elements of a system to be composed of components, attributes, and relationships. Langford (2012, 369) defines a system as, "a bounded, stable group of objects exhibiting intrinsic emergent properties that through the interactions of energy, matter, material wealth, and information provide functions different from their archetypes." The International Council on Systems Engineering (INCOSE) offers two more definitions of system: "a combination of interacting elements organized to achieve one stated purposes" and "an integrated set of elements, subsystems, or assemblies that accomplish a defined objective" (INCOSE 2010, 5).

Overall, these definitions reveal the common theme of connecting functions to form functions that are different when combined in the whole system. Systems can then be connected or integrated to other systems to form a more complex "system of systems" as shown in Figure 1.

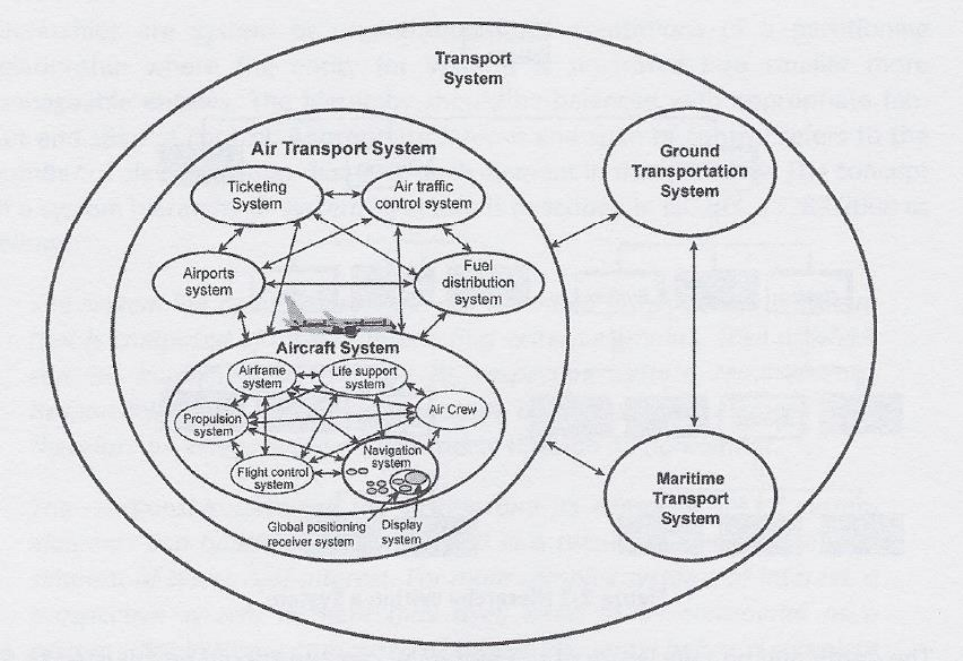

Figure 1. System of systems example for an aircraft system (from INCOSE 2010).

2. Systems Engineering

Systems engineering is a relatively new discipline and its emergence appears in the early to mid-1900s. According to INCOSE, systems engineering originates sometime in the 1930s. In 1937, a British multi-disciplinary team was established to analyze the air defense system (INCOSE 2010). In the United States, systems engineering as a discipline gained recognition during the missile program development during the 1950s (Langford 2012). Like those for systems, the definitions of systems engineering vary.

The *INCOSE Systems Engineering Handbook* references multiple definitions and develops its own:

> Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost, and schedule, performance, training and support, test, manufacturing, and disposal. Systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs. (INCOSE 2010)

Langford (2012, 370) defines systems engineering as:

> The character of systems engineering is to create and express ideas and integrate components into systems that are referred to as products or services. The essence of systems engineering is to unbound the seemingly bounded, broaden the concepts to beyond recognition, open the solution domain to include the ridiculous, and consider the issues and problems in an abstract space rather than as they are posed or presumed to be real. No other discipline or field carries with it that worldview.

These two definitions share some common themes. Both definitions define systems engineering as the broad overview of the development of the entire system, reaching out beyond typical engineering development boundaries. Systems engineering embodies thinking about the broader concerns of the system including the customer needs, the project management concerns, the system life cycle, and the integration of the system with other systems including its environment.

3. Systems Architecting

Capturing a consistent, single definition of systems architecting is as difficult as the attempts for defining system and systems engineering. While systems engineering has been around long enough to acquire its own definitions and recognition as a discipline, systems architecting is still struggling to distinguish itself from the general architecture field. Thus, traditional architecture definitions (the building of structures) have been extended to cover the architecture of systems. Maier and Rechtin (2009) compare the Webster's dictionary definition of architecture to their definition, pointing out that the dictionary definition of architecture tends more towards describing the profession. These definitions are:

- Webster's Definition: "The art or science of building; specifically the art or practice of designing and building structures and esp. habitable ones."
- Maier and Rechtin's Definition: "The structure – in terms of components, connections, and constraints – of a product, process, or element."

Langford defines architecture as the "conceptual and logical structures of objects and processes (and their logical derivatives, e.g., functions or procedures, respectively)" (Langford 2012).

Finally, DODAF Version 1.0 offers the following systems architecture definition, also depicted graphically in Figure 2, "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" (Department of Defense, Deputy Chief Information Officer 2004, ES-1).

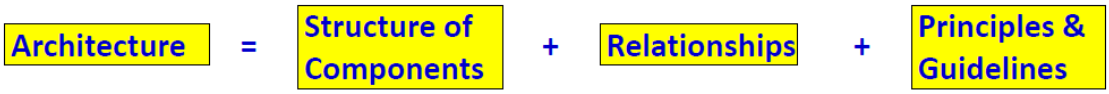

Figure 2. Diagram of systems architecture definition (from Vaneman 2014, 3-3-6, 12, 14-15)

Naturally, all three definitions include "structure" as a key element of architecture. Another common theme is the requirement for "connections" or "relationships." Both Maier and Rechtin's and Langford's definitions classify "process" as part of the structure, which is an important differentiator in defining systems architecture vice building architecture. Systems architecture builds the structure of processes to form a system.

4. Systems Architecting and Systems Engineering

Systems engineers and developers are frequently at odds with the system architects. They may believe that the architecture model development is a superfluous function that can be streamlined and created during the systems engineering processes. However, throughout history, there have been many architecture-based disasters that cost human lives but also examples of successful product-based architectures. According to Maier and Rechtin, the engineering of a system is a deductive process focused on things that can be measured and using analytics that are based on mathematics and science. On the other hand, they classify architecting as an inductive process focused mostly on things that cannot be quantitatively measured and relying on heuristics and guidelines developed from experiences. However, systems architecting and its practitioners provide necessary and distinct value towards the development of systems.

A strong and complementary codependence between architecting and engineering throughout the life cycle of systems development is shown in Table 1.

Table 2. The architecting and engineering continuum: characteristics of the roles (after Maier and Rechtin 2009).

Characteristic	Architecting	Architecting & Engineering	Engineering
Situation/goals	Ill-structured Satisfaction	Constrained Compliance	Understood Optimization
Methods	Heuristics Synthesis **Art** and Science	Art **and** Science	Equations Analysis **Science** and Art
Interfaces	Focus on "mis-fits"	Critical	Completeness
System integrity maintained through	"Single mind"	Clear objectives	Disciplined methodology and process
Management issues	Working for client Conceptualization and certification Confidentiality	Working with client Whole waterfall Conflict of interest	Working for builder Meeting project requirements Profit versus cost

Each characteristic listed above describes the focus area of concern for the architect versus that of the engineer. The middle column highlights the areas of common ground between the two disciplines. Warren Vaneman's September 2015 lecture, "Introduction to System Architectures" provides additional insights into the required characteristics and skills of a system architect, specifically citing from Alexander H. Levis and Lee W. Waganhals to argue, "the architect is NOT the systems engineer. The role of the architect and the role of the systems engineer should not be taken by the same person." Table 2 identifies the ideal skills sets and traits of a good systems architect as defined by Vaneman.

Table 3. Skill sets and traits of a systems architect (after Vaneman 2014, 3-3-6, 12, 14-15).

The Architect's Skill Sets	
Creates Vision	
Thinks Differently	Creatively and holistically
Works for client and stakeholders	Balances interests
Defines and allocates requirements	Understand key metrics
Proposes and develops options	Applies creativity in the development of concepts Considers new technologies
Conducts trade off analysis and optimization	
Resolves ambiguity	
Communicates ideas to others	Creates abstractions (models)
Delivers value	

In summary, architecting and engineering skills, concerns, methods, and purposes are different and, yet, complementary.

5. Systems Architecture Approaches

The evolution of systems architecture requires the development of approaches, methods, and tools to support the development of the system architect's models and products. The DOD developed the DODAF to address these needs. In 2004, DODAF adopted as a requirement for systems development by the DOD Chief Information Officer (CIO). The framework describes viewpoints and models that a systems architect may choose based on the requirements of their particular systems project. Conformance criteria exist for DODAF; however, these criteria only provide recommendations for model representations, not the tools or methods to implement them. Industry has developed a variety of architecture tools to support a wide array of model types and methods.

At the NPS, research is ongoing in the development of a formal architecture approach called Monterey Phoenix to be used to model system behaviors and business processes. The MP approach provides software and systems architects a simple event grammar to model their architectures and evaluate behaviors of the system. The purpose of MP is to "specify, then verify and validate the correct behavior of a system" at the time that the system architecture is being developed (Giammarco and Auguston 2015a). By focusing on the system behaviors in the architecture models, MP exposes unintended behaviors in the systems architecting vice systems development phase. Using MP to model their systems, architects can use its simulation capability to generate a robust set of use cases to evaluate the behaviors of their model (NPS 2015).

B. PROBLEM STATEMENT

Conformance, to the maximum extent possible, with the DODAF is expected for architectures developed in the DOD (DOD 2015a, 4-5). To promulgate the value, usage, and acceptance of the MP approach, a case study mapping DODAF models to MP visualizations determines what MP models satisfy the following DODAF conformance criteria from the DODAF Version 2.02 (Department of Defense, Deputy Chief Information Officer 2015, 231):

- The data in a described architecture is defined according to the DM2 concepts, associations, and attributes.
- The architectural data is capable of transfer in accordance with PES (physical exchange specification). (Department of Defense, Deputy Chief Information Officer 2015, 231)

The ability to use MP-generated models to satisfy DODAF guidelines enables DOD architects to use the approach as a single capability to create, document, and communicate their models.

While DODAF tools are available for model documentation, the MP approach provides a simple grammar and tool that enables the system architect to reduce design complexity while quickly and easily exposing architectural flaws prior to implementation. The Monterey Phoenix website defines the main principles and benefits of the approach (Giammarco and Auguston 2015c). As stated on the website, MP

provides an architecture approach with a focus on system behaviors and the interactions between these behaviors. System behaviors and interactions are modeled separately allowing MP to automatically generate an extensive set of use cases to identify system behaviors (Giammarco 2015). This set of use cases provide the modeler the ability to determine visually the behaviors of the system either intended or unintended. In addition to the automatic generation of use cases, other advantages of the MP approach, as described on the MP website, include early assessment of non-functional requirements, early identification of design flaws with the potential to save costs on rework for errors found during implementation, support of reusable architectural patterns, ability to integrate with standard notations such as UML and SysML, and the simplicity of the MP grammar.

C. RESEARCH QUESTIONS

The thesis explores the following research questions:

1. What DODAF viewpoints and models can be derived using the MP architecture description approach and language?

2. What visualizations could be added to the MP prototype, MP Analyzer, to enhance usage of the MP approach?

3. Can DODAF views and models be used to demonstrate the strength of MP to expose high level design errors and unintended system behaviors?

   (a) Can an MP Application Programming Interface enable the MP technology to be used with existing architecture tools to leverage their feature sets?

   (b) Can a fused product be created?

D. SCOPE AND OBJECTIVE

The scope of this research is to apply the MP approach to existing DODAF models to determine what models can be generated within MP and its prototype tools, MP Analyzer or Eagle6. The objective is to define methods for converting DODAF models into MP to leverage the strength of MP capabilities in exposing unintended system behaviors during the systems architecture phase.

This research uses DODAF models developed for the Joint Training Enterprise Architecture (JTEA) as a case study where available; otherwise, example models are developed. Initial research in converting a JTEA business process modeling notation (BPMN) model (DODAF Operational View-6c) into MP and executing it in the initial MP prototype, Eagle6, showed the correlation of data between the two modeling approaches can be used to generate like graphical views. Further refinement of this initial model expands the BPMN constructs for further correlation of the two approaches. Each DODAF model is evaluated for potential conversion and generation in MP using established criteria.

E. BENEFITS OF STUDY

The thesis research benefits the MP project by determining which DODAF model visualizations can be generated using the MP approach and grammar. Results from this research will inform implementation of visualizations in the MP prototypes, the MP Analyzer and Eagle6. The ability of the MP approach to show alignment with existing and widely used and accepted visualizations will enhance adoption and usage of MP. The advantages of using MP will benefit the broader systems engineering community by providing the ability to specify, verify, and validate system behavior during the architecture-modeling phase of systems development effort (NPS 2015). Conversely, DODAF models developed using other tool sets can be transformed into the MP grammar using the methods discovered in this research in the near term.

II. DEPARTMENT OF DEFENSE ARCHITECTURE FRAMEWORK

A. HISTORY

The DOD has been struggling with successful development, integration, and interoperability of its systems for more than 30 years. An early example of systems interoperability was the inability of the United States to locate Soviet Union SCUD missiles in 1991 during the Gulf War (Dam 2014). Because of the continuing examples of system failures, the DOD formed the Command, Control, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Working Group (C4ISR AWG) to develop a framework for developing architectures. The intent of the framework was to provide guidance for the development of systems that were interoperable and cost effective (Sowell 2006). The C4ISR Architecture Framework Version 1.0 was developed in June 1996 and version 2.0 was completed in December of 1997. The C4ISR Architecture Framework defined three architecture views: operational, systems, and technical (Sowell 2006). By February 1998, the Under Secretary of Defense (Acquisition and Technology), the Acting Assistant Secretary of Defense (C3I), and the Joint Staff Director of C4 Systems (J6) issued a memorandum mandating the C4ISR Architecture Framework, Version 2.0 be used in all C4ISR or associated architectures. However, this memorandum provided guidance only and was only valid for six months (Dam 2014).

The next major change to the architecture framework came in February 2004 with the release of DODAF 1.0. Highlights of the changes in this framework included covering all of DOD not just C4ISR, a more data-centric vice product centric approach, flexibility in product selection based on the architecture problem, the emphasis on capabilities rather than requirements, and the inclusion of "Unified Modeling Language (UML)-like" diagrams (Dam 2014). DODAF 1.0 had four views: all views, operational, systems, and technical.

DODAF 1.5, released in April 2007, introduced new architectural concepts based on the emergence of service-oriented architectures. A key change in this version was the

modification of "systems views" to "systems and services" views. In May 2009, DODAF 2.0 was released changing "views" to "viewpoints, "products" to "models," separating "system and services" views to "system viewpoints" and "service viewpoints," and adding three additional viewpoints (for a total of eight viewpoints): capability, data and information, and project (Dam 2014).

DODAF 2.02 was released in August 2010 and there have been three incremental updates with the latest one being DODAF 2.02 Change 1 released in January 2015. Some changes in this release include updating the "DODAF conformance to four levels (conceptual, logical, physical, and semantic)," technical edits to model descriptions, simplification of information resource flows and associations, and refinement on the meaning of "services," and various clarifications and corrections. For a full listing of the changes, please see the DODAF Version 2.02, Change 1, Volume 1: *Overview and Concepts, Manager's Guide* (DOD 2015a, 4–5). Figure 3 summarizes the DOD timeline for the development of the architectural frameworks and the Clinger-Cohen Act, which set the stage for enforcement of the development and maintenance of system architectures (DOD 2015a, 4–5).

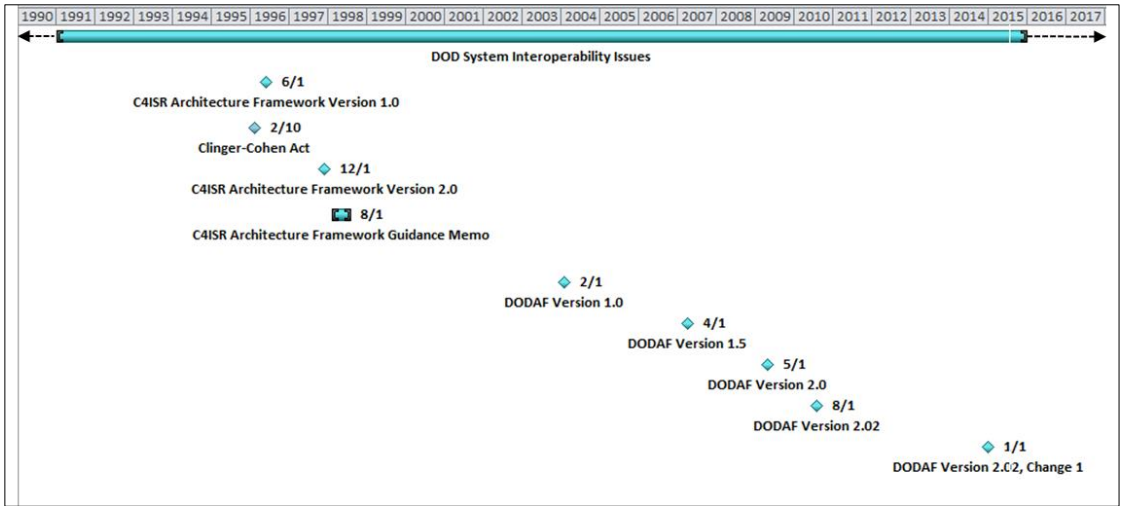

Figure 3. DODAF historical timeline.

As DODAF approaches its 20th anniversary, DOD project managers, system engineers, and architects continue to struggle to develop systems successfully. Architecting is still in its infancy in comparison with other engineering disciplines. As such, the best methods and approaches for architecting today's complex systems have yet to be discovered. DODAF has 51 different models that projects may or may not use that can be implemented in large number of standards, formats, and tools by a workforce with varying levels of expertise. Almost as in a perfect storm, when all these factors combine, a complex and large number of architecture models are built, shoved into DOD required documentation, and then put on the shelf and never used. To improve the art and science of architecting, methods for simplifying and unifying architecture methods and approaches must continue to evolve.

1. Authority

The C4ISR Architecture Framework was originally released with a memorandum that provided for usage of the framework as guidance (Sowell 2006). As such, DOD agencies were not required to develop C4ISR architectures for systems development efforts. In 1996, the Clinger-Cohen Act was passed to transform the acquisition and management of IT by requiring more rigor and structure in the processes. The DOD Chief Information Officer must provide the oversight required to ensure that all IT systems are "interoperable, secure, properly justified, and contribute to mission goals" (DAU 2015). The DODAF allows the DOD Chief Information Officer to support this law, follow the guidance from the Office of Management and Budget (OMB), and align with DOD directives and instructions (DOD 2015a, 4–5).

2. Conformance

With the authorities provided, the DOD CIO expects conformance to the DODAF by DOD components. While the DODAF is developed to be "fit-for-purpose" allowing sensible flexibility in development of models that meet the architecture at hand, DOD components must conform to the "maximum extent possible" with the DODAF. This conformance permits the purpose and objectives of the architecture framework to be realized: reuse of information, sharing of architecture artifacts, models and viewpoints,

and common understanding of the architecture. Conformance is said to be achieved when the following two items are met:

- Architecture data is defined using the DODAF Meta Model (DM2) including the concepts, associations, and attributes
- Transference of the data in accordance with the PES (physical exchange specification) is achievable (DOD 2015a, 4–5).

B. OVERVIEW: VIEWPOINTS AND MODELS

DODAF 2.02 organizes the framework as models and viewpoints. The models are the artifacts (diagrams or documents) which describe the aspects of the target architecture within the viewpoints. The framework allows DOD architects to develop to a set of standards providing the ability to exchange data among differing system architectures. DODAF does not dictate the methods and techniques system architects will use; however, the data created must conform to the DM2 (Department of Defense [DOD] 2015b, 10). There are eight viewpoints as shown in Figure 4.

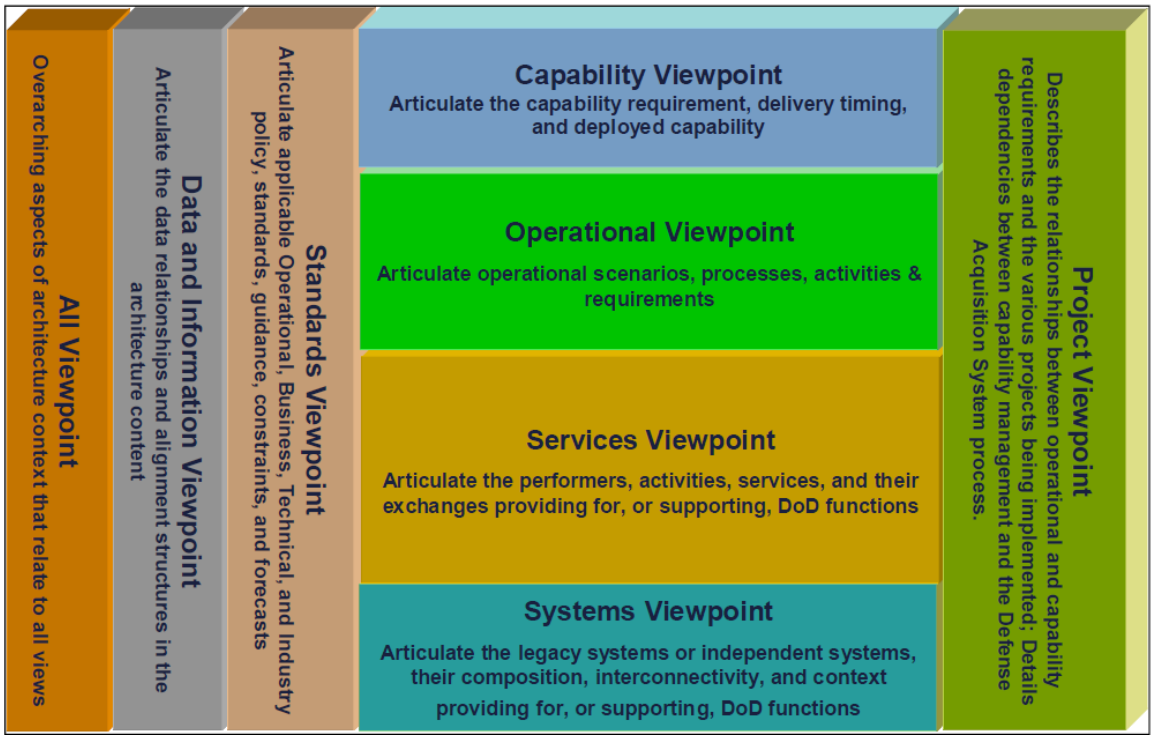

Figure 4. DODAF has eight viewpoints (from DOD 2015b, 10).

A total of 51 models exist within the eight viewpoints. Architects select the viewpoints and models to develop based on the purpose of their architecture. Additionally, architects select the methods, techniques, and tools for developing the architecture artifacts (DOD 2015b, 10). Within the scope of a particular architecture effort, the overall project/engineering management teams need to follow the same methods and techniques and use the same tools to realize the benefits of an integrated architecture. Tables 3–6 describe each model within its viewpoint and the typical representation used to describe the model.

Table 4. Representations for DODAF 2.02 all, capability, and data/information viewpoints (Department of Defense, Deputy Chief Information Officer 2015, 231), (DOD 2015b, 10), and (Dam 2014).

Model	Name/Description	Typical Representation
All Viewpoints		
AV-1	Overview and Summary Information - describes a project's visions, goals, objectives, plans, activities, events, conditions, measures, effects (outcomes), and produced objects.	Structured text document.
AV-2	Integrated Dictionary - an architectural data repository with definitions of all terms used throughout the architectural data and presentations.	Data hierarchy, text definition with source reference.
Capability Viewpoints		
CV-1	Vision - the overall vision for transformational endeavors, which provides a strategic context for the capabilities described and a high-level scope.	Textual descriptions and relationship diagrams.
CV-2	Capability Taxonomy - a hierarchy of capabilities that specifies all the capabilities that are referenced throughout one or more architectural descriptions.	Structured/hierarchical list or chart.

Model	Name/Description	Typical Representation
CV-3	Capability Phasing - the planned achievement of capability at different points in time or during specific periods of time.	Table with rows representing capabilities (from CV-1) and columns representing phases (from CV-2) or timeline. Can be represented graphically.
CV-4	Capability Dependencies - the dependencies between planned capabilities and the definition of logical groupings of capabilities.	Graphical using connecting lines or matrix.
CV-5	Capability to Organizational Development Mapping - the fulfillment of capability requirements shows the planned capability deployment and interconnection for a particular Capability Phase. The CV-5 shows the planned solution for the phase in terms of performers and locations and their associated concepts.	Table with rows/columns representing capabilities and organizations. Can be represented graphically.
CV-6	Capability to Operational Activities Mapping - a mapping between the capabilities required and the operational activities that those capabilities support.	Table with rows/columns representing capabilities and operational activities. Can be represented graphically.
CV-7	Capability to Services Mapping - a mapping between the capabilities and the services that these capabilities enable.	Table with rows/columns representing capabilities and services. Can be represented graphically.
Data and Information Viewpoints		
DIV-1	Conceptual Data Model - the required high-level data concepts and their relationships.	Graphical representation to depict data concepts and relationships.

Model	Name/Description	Typical Representation
DIV-2	Logical Data Model - the documentation of the data requirements and structural business process (activity) rules.	Graphical representation using appropriate data modeling methodology for system.
DIV-3	Physical Data Model - the physical implementation format of the logical data model entities, e.g., message formats, file structures, physical schema.	Graphical representation using appropriate data modeling methodology for system.

Table 5. Representations for DODAF 2.0 operational and project viewpoints (Department of Defense, Deputy Chief Information Officer 2015, 231), (DOD 2015b, 10), and (Dam 2014).

Model	Name/Description	Typical Representation
Operational Viewpoints		
OV-1	Operational Concept - the high-level graphical/textual description of the operational concept.	Graphical representation of the architecture. Typically done in a Microsoft PowerPoint document and must include a textual description.
OV-2	Organizations and Resources - a description of the resource flows exchanged between operational activities. Shows a need to exchange information. Can also show flows of funding, personnel, and materiel.	Graphical representation of using arrows to depict operational needlines and resource flows.
OV-3	Operational Resource Flow Matrix - a description of the resources exchanged and the relevant attributes of the exchanges.	Table representation.
OV-4	Organizational Relationships Chart - the organizational context, role or other relationships among organizations. Used to show roles or actual organizational constructs.	Graphical organizational charts.

Model	Name/Description	Typical Representation
OV-5a & OV-5b	Operational Activity Decomposition Tree - the capabilities and activities (operational activities) organized in a hierarchal structure. Operational Activity Model - The context of capabilities and activities (operational activities) and their relationships among activities, inputs, and outputs. Additional data can show cost, performers, or other pertinent information.	Activity modeling methodology of choice, such as, IDEF0 (Integration Definition for Function Modeling) or class diagrams.
OV-6a	Operational Rules Model - one of three models used to describe activity (operational activity). It identifies business rules that constrain operations.	Statements written in natural language.
OV-6b	State Transition Description - one of three models used to describe operational activity (activity). It identifies business process (activity) responses to events.	Graphical representation based on the state chart diagram. NOTE: helps to identify behavioral errors as noted by DODAF itself!
OV-6c	Event-Trace Description - One of three models used to describe activity (operational activity). It traces actions in a scenario or sequence of events.	Graphical representation using event-trace diagram methodology of choice, such as BPMN.
Project Viewpoints		
PV-1	Project Portfolio Relationships – describes the dependency relationships between the organizations and projects and the organizational structures needed to manage a portfolio of projects.	Hierarchical organizational breakdown.
PV-2	Project Timelines - a timeline perspective on programs or projects, with the key milestones and interdependencies.	Graphical representation. Often a Gantt chart.
PV-3	Project to Capability Mapping - a mapping of programs and projects to capabilities to show how the specific projects and program elements help to achieve a capability.	Table representation with rows as capabilities and columns for programs, projects, portfolios, or initiatives.

Table 6. Representations for DODAF 2.0 services and standards viewpoints (Department of Defense, Deputy Chief Information Officer 2015, 231), (DOD 2015b, 10), and (Dam 2014).

Model	Name/Description	Typical Representation
Services Viewpoints		
SvcV-1	Services Context Description - the identification of services, service items, and their interconnections.	Graphical representation.
SvcV-2	Services Resource Flow Description - a description of the resource flows exchanged between services.	Graphical representation.
SvcV-3a	Services-Systems Matrix - the relationships among or between systems and services in a given architectural description.	Table representation.
SvcV-3b	Services-Services Matrix - the relationships among services in a given architectural description. It can be designed to show relationships of interest, (e.g., service-type interfaces, planned vs. existing interfaces).	Table representation.
SvcV-4	Services Functionality Description - the functions performed by services and the service data flows among service functions (activities).	Graphical representation.
SvcV-5	Operational Activity to Services Traceability Matrix - a mapping of services (activities) back to operational activities (activities).	Table representation.
SvcV-6	Services Resource Flow Matrix - provides details of service Resource Flow elements being exchanged between services and the attributes of that exchange.	Table representation.
SvcV-7	Services Measures Matrix - the measures (metrics) of Services Model elements for the appropriate time frame(s).	Table representation.
SvcV-8	Services Evolution Description - the planned incremental steps toward migrating a suite of services to a more efficient suite or toward evolving current services to a future implementation.	Timeline diagram.

Model	Name/Description	Typical Representation
SvcV-9	Services Technology & Skill Forecast - the emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future service development.	Table, timeline, or herringbone diagram.
SvcV-10a	Services Rules Model - one of three models used to describe service functionality. Identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.	Statements written in natural language.
SvcV-10b	Services State Transition Description - one of three models used to describe service functionality. Identifies responses of services to events.	Graphical representation based on the state chart diagram.
SvcV-10c	Services Event-Trace Description - one of three models used to describe service functionality. Identifies service-specific refinements of critical sequences of events.	Graphical representation using event-trace diagram methodology of choice, such as BPMN.
Standards Viewpoints		
StdV-1	Standards Profile – the listing of standards that apply.	Text document.
StdV-2	Standards Forecast - the description of emerging standards and potential impact on current solution elements, within a set of time frames.	Graphical representation. Often a GANTT chart.

Table 7. Representations for DODAF 2.0 systems viewpoints (Department of Defense, Deputy Chief Information Officer 2015, 231) and (Dam 2014).

Model	Name/Description	Typical Representation
Systems Viewpoints		
SV-1	Systems Interface Description - the identification of systems, system items, and their interconnections.	Graphical representation.

Model	Name/Description	Typical Representation
SV-2	Systems Resource Flow Description - a description of the resource flows exchanged between systems.	Graphical representation.
SV-3	Systems-Systems Matrix - the relationships among or between systems and services in a given architectural description.	Table representation.
SV-4	Systems Functionality Description - the functions performed by systems and the system data flows among system functions (activities).	Graphical representation.
SV-5a	Operational Activity to Systems Function Traceability Matrix – a mapping of system functions (activities) back to operational activities (activities).	Table representation.
SV-5b	Operational Activity to Systems Traceability Matrix - a mapping of systems (activities) back to operational activities (activities).	Table representation.
SV-6	Systems Resource Flow Matrix - provides details of system Resource Flow elements being exchanged between systems and the attributes of that exchange.	Table representation.
SV-7	Systems Measures Matrix - the measures (metrics) of systems model elements for the appropriate time frame(s).	Table representation.
SV-8	Systems Evolution Description - the planned incremental steps toward migrating a suite of systems to a more efficient suite or toward evolving current systems to a future implementation.	Timeline diagram.
SV-9	Systems Technology & Skill Forecast - the emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future system development.	Table, timeline, or herringbone diagram.
SV-10a	Systems Rules Model - one of three models used to describe system functionality. Identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.	Statements written in natural language.

Model	Name/Description	Typical Representation
SV-10b	Systems State Transition Description - one of three models used to describe system functionality. Identifies responses of system to events.	Graphical representation based on the state chart diagram.
SV-10c	Systems Event-Trace Description - one of three models used to describe system functionality. Identifies system-specific refinements of critical sequences of events.	Graphical representation using event-trace diagram methodology of choice, such as BPMN.

Each of the DODAF models above is evaluated against defined criteria to determine the feasibility of developing MP code and generating an MP model.

C. COMMUNICATING ARCHITECTURES

Architectures and the associated models are an abstraction of the overall system. Collectively, all the models form the system. In order to effectively communicate an architecture, systems architects must be able to articulate clearly the models to stakeholders for their comprehesion, modification, and approval.

In addition to providing standards, interoperability, potential reuse, discipline in development, and potential cost savings, models communicate the system to its stakeholders. Stakeholders can include senior leaders, users, engineers, developers, testers, project managers, financial managers, safety managers, and politicians. Stakeholders can be internal or external to the developing organization. For example, system interfaces require cooperation and design consideration from the system stakeholders.

Architects use models as tools to communicate the system requirements to the stakeholders for approval, verification, and validation of the system prior to its implementation. Iterative reviews of the models with the appropriate stakeholders provide for early discovery and correction of design issues. The models allow stakeholders to ask “what-if” questions and to identify capability gaps early in the system life cycle. Without the ability to effectively communicate the architecture, architects

cannot deliver the value associated with producing the architecture and verifying its feasibility prior to the cost of implementation. Once implementation begins, correcting architectural issues often requires extensive rework, schedule delays, and cost overruns. It is common for the DODAF artifacts to be produced as part of the checklist compliance and then shelved without further consideration.

In the *Art of Systems Architecting*, Maier and Rechtin (2009, 397) provide a list of system architecture heuristics. They quote the following heuristic, "One person's architecture is another person's detail. One person's system is another's component. (Robert Spinrad, 1989)." This heuristic provides important guidance when communicating with others about a complex or system of systems architecture. For each stakeholder, the architect needs to consider that party's focus and expertise when devising an approach for communicating about the architecture models. For example, every systems team positions their system as the "center of the universe," with everything interacting and/or interfacing with it. When architecting any solution, architects need to understand that there are "mini-architectures" that comprise the whole and impact their system architecture. Stakeholders of the "mini-architectures" can be threatened by the development of a new one because it might duplicate functionality or replace their systems and the communication approach must consider this possibility.

The accompanying prescriptive heuristic is "In order to understand anything you must not try to understand everything. (Aristotle, 4th century B.C.)" (Maier and Rechtin 2009). Aristotle's statement also helps formulate the architecture communication at digestible levels of detail. Why is this important? The DODAF approach is "fit-for-purpose,"— not every viewpoint or model needs to be produced if the value of the model does not fit the architecture under development. Additionally, the models must be readable in forms that are understandable to the respective stakeholder reviewing the model. Despite these guidelines, the many systems development projects continue to create ill-defined, complex, and unverifiable models and represent them as DODAF viewpoints. As presented to the author, November 17, 2014, by Brian Gregg (pers comm), Figure 5 is a joint training system SV-2, a systems resource flow description that, as described in Table 6 above, is a "description of the resource flows exchanged between

systems." An SV-2 is typically represented graphically; however, it should be a model of the resource flows between systems. The complexity and inaccurate representation of this SV-2 makes verification of the exchanged resource flows impossible. For example, in the yellow box, the "Training Event Manager" and the "Joint Exercise Design Tool" provide inputs to a composite grouping of six components. These six components have no outputs and only have inputs coming from other systems. It seems unlikely that none of the six component systems would produce any output to any other component system. Additionally, the model includes performers, network details, and implementation details such as the blue mass behind some of the systems, which represents the intention of these systems to run in the cloud.

The SV-2 model is an abstraction at the systems level of resources produced and consumed. Even at this level of abstraction, it is necessary that the model communicate precision. Any ambiguity found in a high-level model must be corrected for common understanding with the system stakeholders and implementation team. The MP approach provides for early model assessments to discover these types of problems and reduce potential errors before the system design is implemented.

.

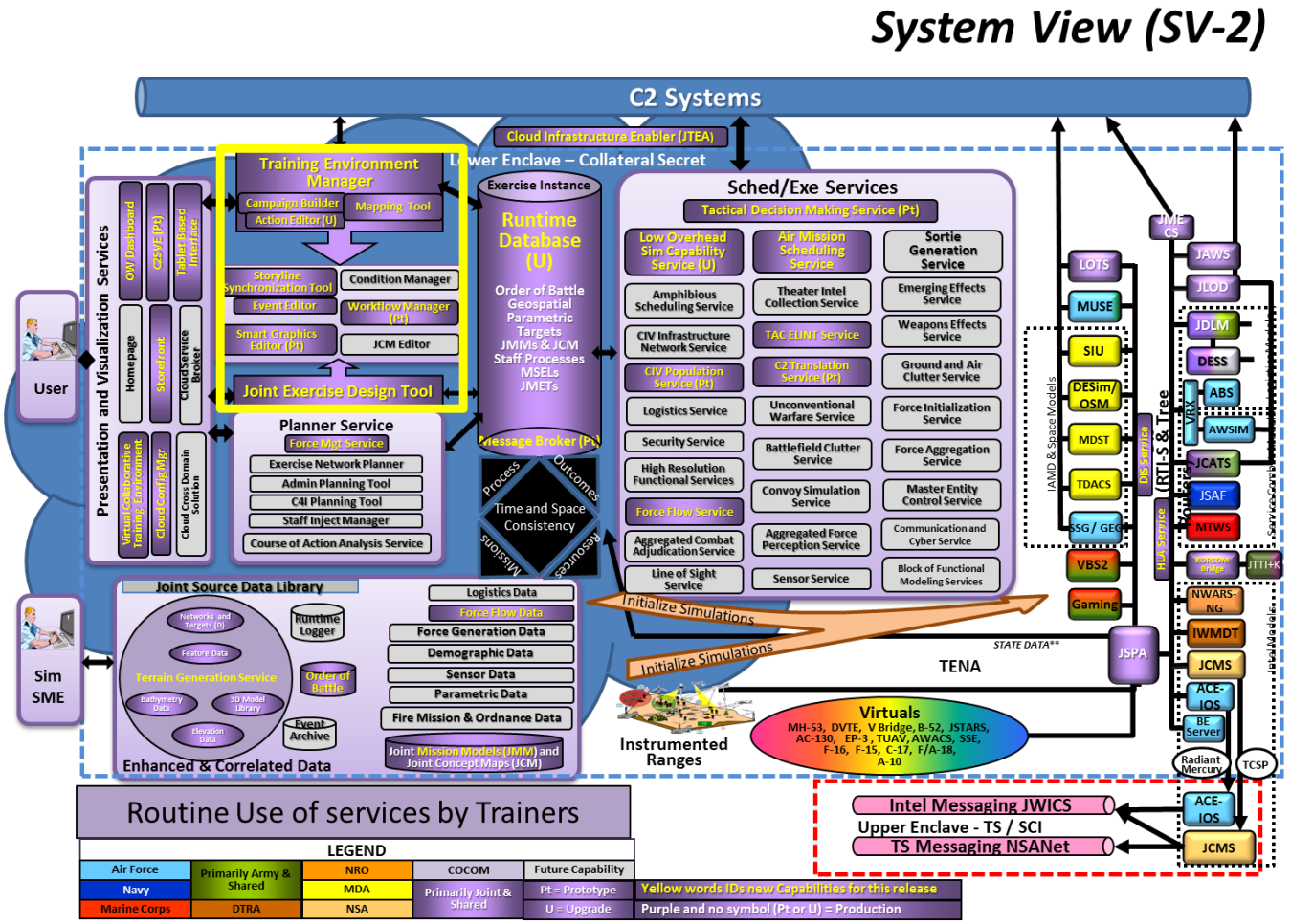

Figure 5. Example of a complex, ill-defined, unverifiable DODAF SV-2.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MONTEREY PHOENIX BEHAVIOR MODELING APPROACH

A. INTRODUCTION

The struggle to improve software and systems development is well known to those entrenched in the field including this author. Despite more than a 50-year history of new methodologies, approaches, and technology advances, software and systems development remains one of the most challenging areas for product deliveries that are of high quality, functionally useful, on time and within cost. Yourdon and Argila (1996), who championed the object oriented analysis and design methods, highlight the history of software development methods starting with the 1960s waterfall approach that provided formal methods and processes for developing software. The waterfall approach was an attempt at preventing ad hoc and chaotic software development. By the 1970s, the waterfall approach fell out of favor because of the massive amount of documentation required to proceed to the next phase and the need for quicker development cycles (Yourdon and Argila 1996).

In the late 1970s, model-based software engineering was introduced in Tom Demarco's (Demarco 1979) book, *Structured Analysis and System Specification*. According to Demarco, complex software systems should be developed by first creating models prior to spending resources to implement them (Yourdon and Argila 1996). Almost forty years later, the software and systems development community is still working to apply this concept to improve software development.

By 1996, object oriented analysis and design emerged as the newest methodology for developing software systems. According to Ed Yourdon (Yourdon and Argila 1996) in his book, *Case Studies in Object Oriented Analysis and Design*, the principle of separation of concerns is an important heuristic for model-based software engineering. Yourdon applies this to the separation of the analysis and design models and this heuristic is the foundation for his approach (Yourdon and Argila 1996).

Almost ten years ago, service oriented architecture (SOA) hit the information technology stage with more promises of improving software and system development. SOA claimed to provide an architectural model intended to provide organizations agility and cost-effectiveness against the ever-growing liability of information technology requirements (Erl et al. 2009). Grady Booch notes the following in the forward to Thomas Erl's *SOA Design Patterns* book:

> The entire history for software engineering can be characterized as one of rising levels of abstraction. We see this in our languages, our tools, our platforms, and our methods. Indeed abstraction is the primary way that we as humans attend to complexity-and software-intensive systems are among the most complex artifacts ever created. (Erl 2009, xxxvii)

Another recent trend in software development is the agile methodology using the scrum process. Agile development uses collaborative, flexible, and iterative methods to accelerate software development timelines and improve outcomes (INCOSE 2010). Agile development is paired with the scrum framework, which focuses on complete visibility of the software development processes (Schwaber 2004). Once again, the software industry introduced an approach targeted to improve, control, and manage software and systems development. Ken Schwaber, a co-developer of the scrum process in the early 1990s, states that "complex problems are those that behave unpredictably" (Schwaber 2004). In *Agile Project Management with Scrum*, Schwaber (2004) discusses that scrum is based on the idea that software should be developed using empirical process control rather than defined process control—a process that repeatedly produces results of acceptable quality. The implementation of empirical process control has three tenets: visibility, inspection, and adaption, which are applied to code development (Schwaber 2004).

No one has yet to find a magical combination of processes, framework, and tools that can be applied to software and system development efforts to deliver high quality products, within schedule, and within budget. In the "No Silver Bullet*"* article, Brooks (1987) laments that in the upcoming decade there is "no silver bullet." He specifically states, "There is no single development, in either technology or in management technique, that by itself promises even on order-of-magnitude improvement in productivity, in reliability, in simplicity" (1987, 1). Naval Postgraduate School

researchers have merged the concepts of model-based software engineering, heuristic based architecture approaches, formal methods, and the use of abstraction to reduce complexity, and system behaviors to develop the MP approach. While MP is no "silver bullet," MP's simplified set of concepts offers a new lens for managing current system complexity.

Per the MP home page, "Monterey Phoenix is a formal architecture description approach and language for system behavior and process modeling" (NPS 2015). The MP approach uses a simple event grammar to model software and systems architectures. In MP, the systems architect models the behaviors for each component and the interactions between the components are modeled separately as shown in Figure 6. Auguston (2014) states that, "behavior modeling is the core" of the MP approach. This separation of the system behaviors from the system interactions enables the MP approach to automatically generate use case scenarios for human inspection and reveal the unintended behaviors to the systems architect in a manageable and readily understandable form (Auguston et al. 2012, 1).

MP's approach to center on the systems behaviors and the interactions among them enables simplicity of the model by reducing the concepts required to create it (Auguston et al. 2012, 1). DODAF, for example, has 51 defined models that can be developed in the method and tool of choice, creating a large number of architectural concepts, notations, and models to understand and manage across systems. As defined by its authors, MP, on the other hand, has only three constructs: events that represent activities executed in the system, and two relationships between events: inclusion and precedence.

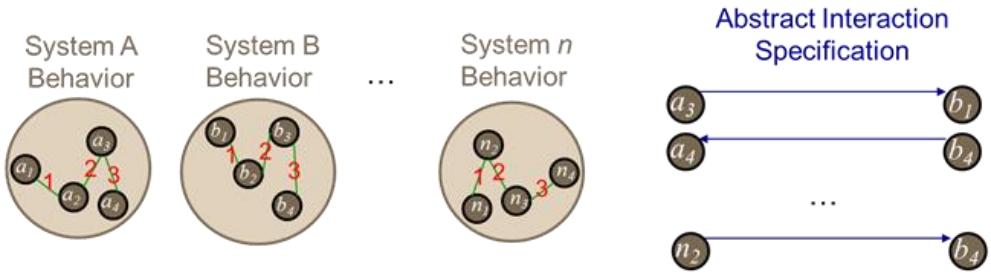

Figure 6. Separation of system behaviors and system interactions. Events/activities are shown as a_1, b_1, n_1, … a_4, b_4, n_4 (from Giammarco, Farah-Stapleton, and Auguston 2014).

The guiding principles and advantages of the MP approach change the thought paradigm most software and systems architects have learned about developing systems. The first principle is the concept of behavior as the basis for system and software architecture modeling, rather than modeling components of functionality and the data flows between them. This approach guides systems developers to consider behaviors and their interactions (NPS 2015).

The second principle focuses on the importance of the environment behavior in systems architecture. In MP, there is a single method for modeling the entire system including its software, hardware, and business processes. This principle enables MP to generate automatically use cases for inspection and validation exposing possible unintended behaviors in the interactions with the environment. MP generates the event traces based on Daniel Jackson's small scope hypothesis that asserts, "if the analysis considers all small instances, most flaws will be revealed" (Jackson 2012, 15).

MP provides for executable architectures for assessment through its automatic generation of use cases, which provides early review of technical requirements. In the future, MP will include the capability to assess performance, latency, and throughput. The automatic generation of the use cases provides formal verification and validation PRIOR to the start of system implementation. Discovery of potential errors or unintended behaviors early in the system development life cycle can reduce project costs and prevent schedule delays. The MP prototype, MP Analyzer, generates multiple architectural views for improved communications with stakeholders with varying perspectives.

Finally, MP fosters reuse of models and is developed to be integrated into existing industry languages and frameworks, such as UML, SysML, and DODAF. Figure 7 maps the DODAF meta-model (DM2), unified profile of DODAF and MODAF (UPDM), and the Lifecyle Modeling Language (LML) concepts to MP concepts (Giammarco and Auguston 2015c).

DM2, UPDM Concepts		LML Concepts		MP Concepts	
Activities / Functions…	1. are "activity" if operational; " function" if mechanized. 2. are decomposed into other activities / functions, respectively. 3. are placed in sequence by position on a branch or by triggering with specific resources (inputs and outputs).	Actions…	1. are activities or functions. 2. are decomposed into other actions. 3. are placed in sequence by position on a branch or by triggering with specific resources (inputs and outputs).	Events…	1. may be typed as action, activity, or function as needed. 2. *include* other events. 3. *precede* other events; inputs and outputs are not separate concepts.
Performers / Components…	1. are decomposed into other performers / components, respectively. 2. perform the activities.	Assets…	1. are decomposed into other assets. 2. perform the actions.	Root Events…	1. are used to represent the performer, component, or asset. 2. include events performed by the performer, component, or asset.
Resources…	1. represent information, data, or matter/energy flows. 2. can be used to impose a sequential order on activities or functions (as triggers).	Input / Outputs…	1. represent information, data, or matter/energy flows. 2. can be used to impose a sequential order on actions (as triggers).	Events…	1. represent the operations performed on data, information, matter or energy; input/outputs are not a separate concept. 2. are sequentially ordered using the *precedes* relation.
Needlines / Links…	1. represent connections between performers / components, respectively. 2. transfer items.	Conduits…	1. represent connections between assets. 2. Transfer input/outputs.	Interactions…	1. represent connections between events (from which connections among assets (root events) can be derived). 2. are used to synchronize events occurring in different systems (root events) or parts of a system. 3. specify precedence relations between events in different systems or parts of a system.

Figure 7. Mapping of MP concept s to DM2 UPDM, and LML concepts (from Giammarco and Auguston 2015a).

B. LANGUAGE

A key advantage of the MP approach is the simplicity of its grammar that revolves around the concept of events. Behavior is represented as a set of events and the two relationships of inclusion and precedence. Precedence allows modeling of dependency, and inclusion provides for decomposition. The MP website provides a basic overview of an event grammar rule and the associated event patterns as shown in Figure 8 and Figure 9.

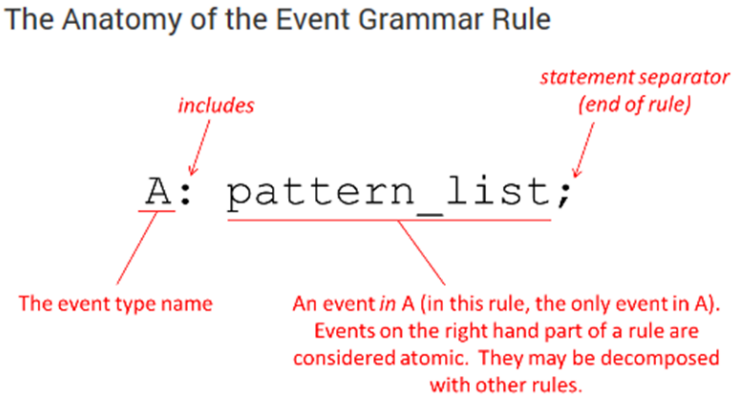

Figure 8. MP event grammar rule (from Giammarco and Auguston 2015b).

Natural Language Description of Pattern	Pattern Expressed as MP Event Grammar Rule
Ordered sequence of events (B followed by C)	A: B C;
Alternative events (B or C)	A: (B | C);
Optional event (B or no event at all)	A: [B];
Ordered sequence of zero or more events B	A: (* B *);
Ordered sequence of one or more events B	A: (+ B +);
Unordered set of events B and C (B and C may happen concurrently)	A: {B, C};
Unordered set of zero or more events B	A: {* B *};
Unordered set of one or more events B	A: {+ B +};

Figure 9. MP event patterns.

A detailed specification of the language is available in Behavior Models for Software Architecture report (Auguston 2014) and in the Monterey Phoenix System and Software Architecture Modeling Language report (Auguston 2015). This thesis provides a brief discussion of the language to provide the reader basic comprehension of how the MP models are generated for consideration as DODAF models.

MP generates instances of event traces from the grammar rule, which are visualized to show two types of edges: precedence and inclusion. Boxes represent events, dotted line arrows represent inclusion, and solid line arrows represent precedence (Auguston 2014). Figure 11 and Figure 12 show these representations as created in the MP prototypes.

The MP language uses the ROOT construct to define a root event and the grammar provides for definition of the set of events included in the root event. For example, the following code from the SV-2 model defines the root event *C2_Systems* that includes two ordered events: *send_C2_data* and *get_TEM_data*.

```
ROOT C2_Systems:
          send_C2_data
          get_TEM_data;
```

The COORDINATE operation defines the interaction (behaviors) between root events by using the PRECEDES relation in a loop (DO / OD) for the set of behaviors. The following code from the SV-2 model first defines another root event, *TEM*. Next the COORDINATE operation models the interaction between the *C2_Systems* and the *TEM: send_C2_data* and *get_C2_data.*

```
ROOT TEM:
          get_C2_data
          send_TEM_data
          get_vis_data
          get_RDS_data;
COORDINATE
     $x: send_C2_data    FROM C2_Systems,
     $y: get_C2_data     FROM TEM
     DO ADD $x PRECEDES $y; OD;
```

The event grammar provides the ability to develop common architecture patterns to develop robust models as shown Figure 10.

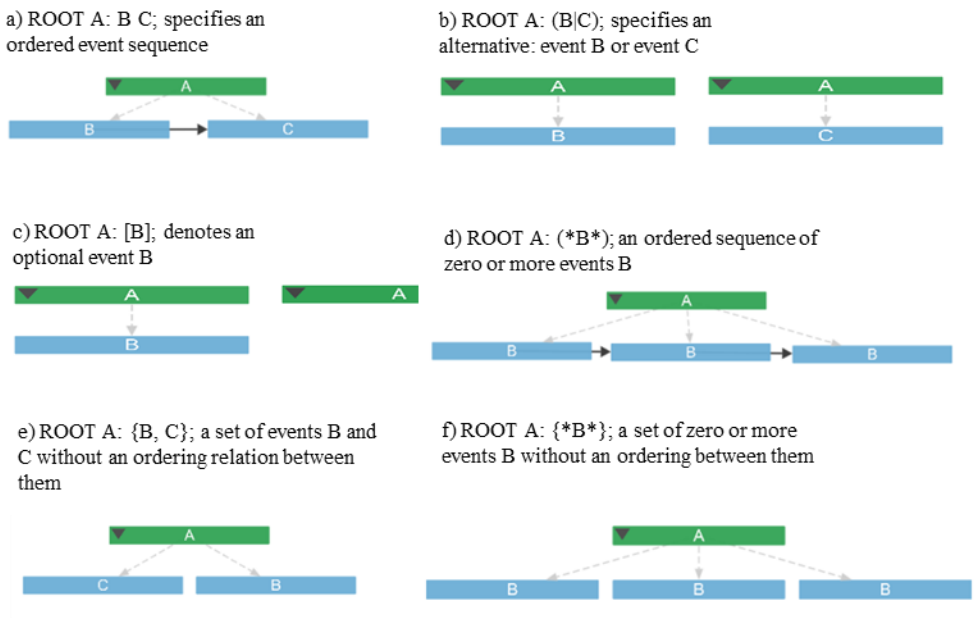

Figure 10. MP event patterns and sample event traces (Auguston 2014).

C. MP PROTOTYPES

Initially, a beta tool (Eagle6) was developed to process the MP language and generate a single graphical display which required manual manipulation of the graph to view (November 2014). This initial visualization in Eagle6 was difficult to view, modify, and to use as a communication platform**.** As of July 2015, Eagle6 generates two organized visualizations with horizontal and vertical orientations as shown in Figure 11 (Rivera 2009).The Rivera Group and NPS have executed a cooperative research and development agreement (CRADA) to share NPS source code and update their Eagle6

implementation.

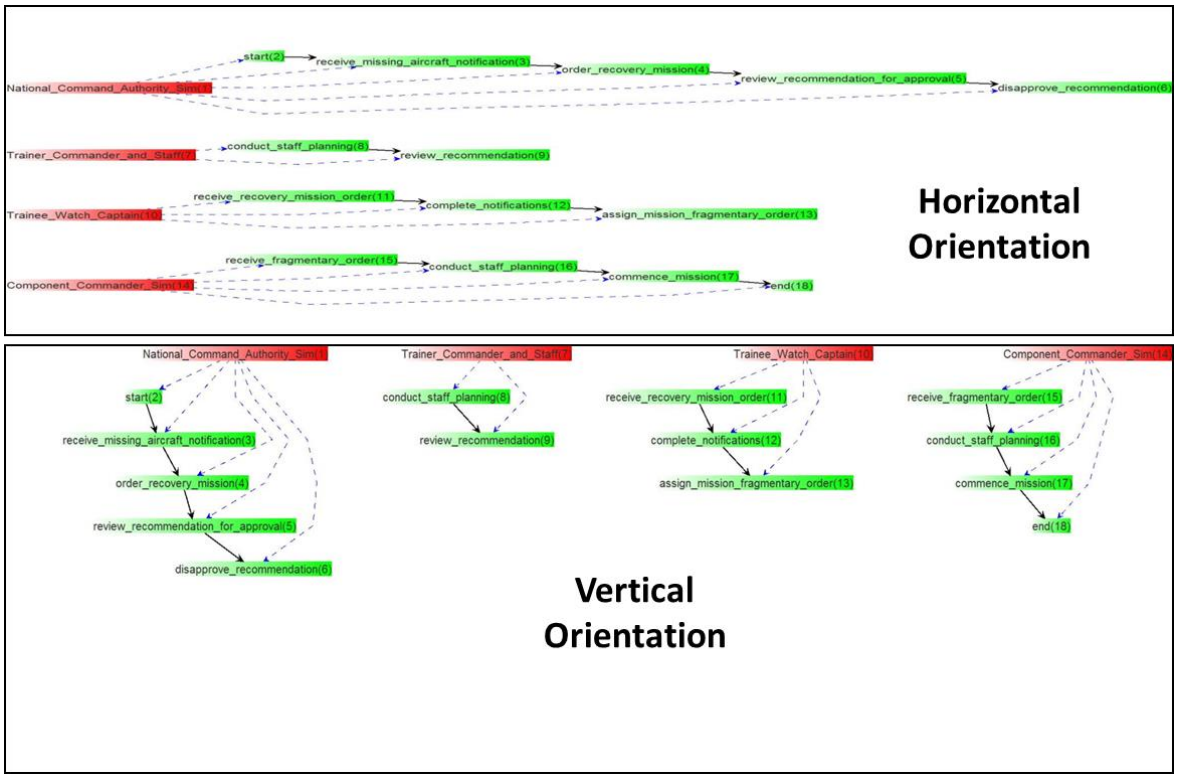

Figure 11. Horizontal and vertical orientation graphs generated from Eagle6, at the time of this writing.

The MP project is developing a new tool on a public server called the MP Analyzer which provides an integrated development environment for MP code development and three visualizations: force, sequence, and swim lanes as shown Figure 12. The DODAF models evaluated for this thesis were generated using only the MP Analyzer prototype.

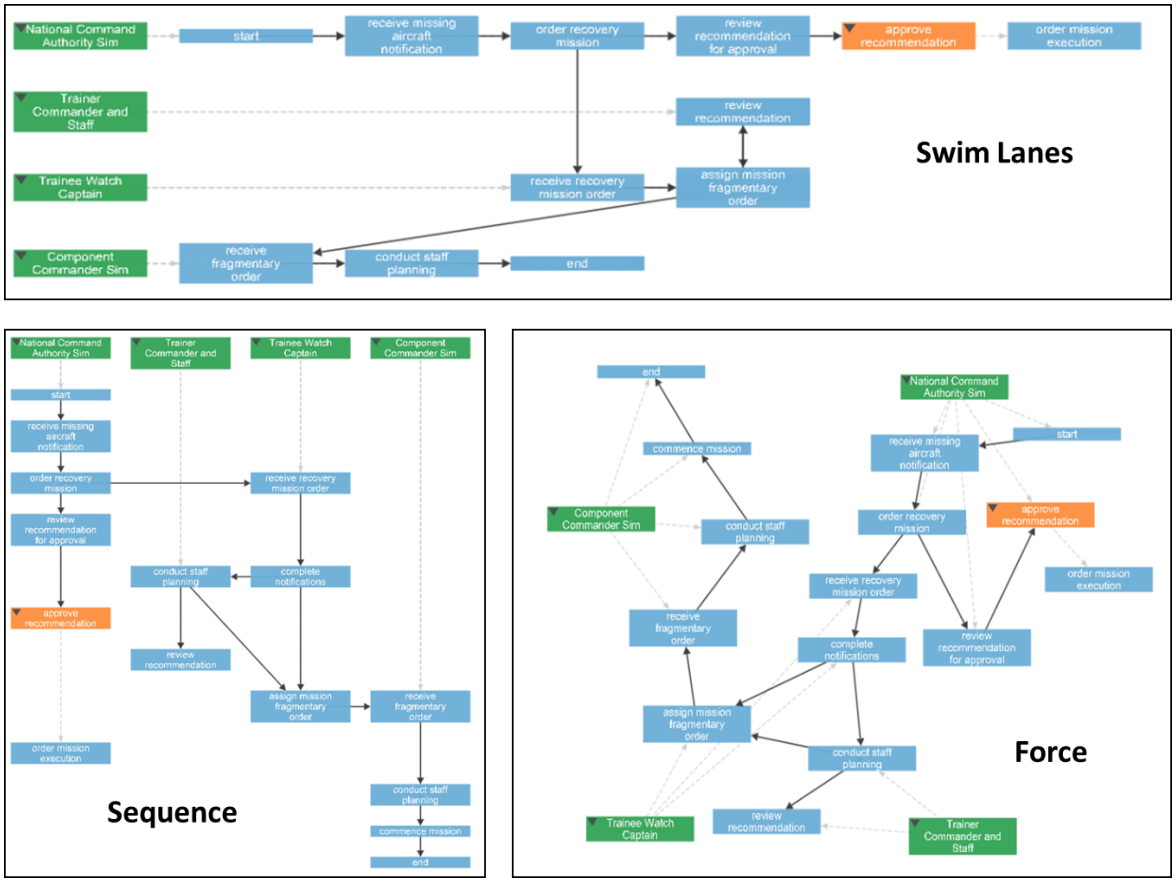

Figure 12. Visualization types available in the MP Analyzer using an example MP model.

The initial MP Analyzer prototype delivers good functionality for developing MP models. Basic functionality includes a code editor, a compiler status viewer, graphics viewer, and a navigation panel shown in Figure 13. The MP Analyzer high-level functions include:

- ability to view only the code or graphs or to split the view
- ability to import from a set of MP examples or your own MP code base
- ability to export MP code, graphs, and complete models (as file types *.mp or *.wng)
- ability to increase the scope of your run to generate even traces
- ability to view the compile results including errors
- ability within the editor to highlight syntax errors during input

- ability to generate, zoom in/out, and manipulate three graph types: swim lanes, sequence, and force

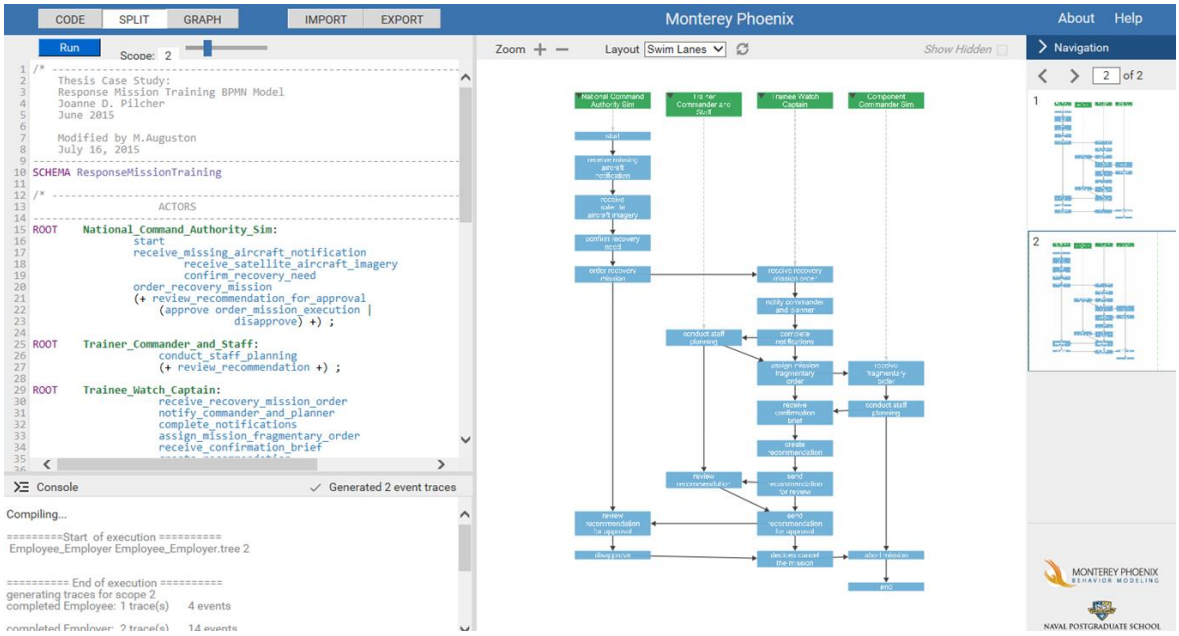

Figure 13. Screenshot from MP Analyzer prototype tool.

MP is an emerging capability designed to shift the way modelers think about developing system architectures by focusing on the behaviors of the system and their interactions. The ability to generate DODAF models from the MP prototype provides near term DODAF compliance for DOD projects and allows modelers to realize the early benefits of the current MP capabilities. The MP effort is planning on future extensions that will bring tighter alignment with DODAF models such as event attributes and assertion checking. Table 7 shows other planned MP enhancements as provided by Giammarco. (pers comm.)

Table 8. Future planned MP enhancements.

MP Enhancements	**MP Analyzer Enhancements**
Assertion checking and queries for filtering for particular traces of interest	Hover over a box or interaction to view attributes (when attributes have been implemented)
Event attributes such as duration and probability	Right-click menu for boxes to perform actions, like upload a picture icon for the box

MP Enhancements	MP Analyzer Enhancements
Ability to insert notes in the code that will print to the diagram (for debugging purposes)	Click box, highlight corresponding events in the code
Static model checking	Click arrow, highlight corresponding interactions in the code
Enhanced error reporting and feedback for debugging	Ability to customize boxes and arrows by color, line weight and style (e.g., dashes, dots, solid light, solid heavy)
	Experimental views, such as timelines and Gantt Charts, and three dimensional and fish-eye navigation of roots and traces
	Standard views, such as SysML state charts and activity models (may be left for commercial implementations)

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CASE STUDY

A. INTRODUCTION

The DOD faces a long-term, resource-constrained environment that creates challenges in providing relevant and realistic training to the warfighter in the current technical environment. Continuing with the delivery of large-scale, resource- and people-intensive, and centrally located joint training is unsustainable. The joint training community is addressing this challenge through an enterprise architecture effort, which modernizes the joint training environment. The current joint training architecture is a complex, highly federated, and manually integrated system of systems including programs of record, government off-the-shelf (GOTS), and commercial off-the-shelf (COTS) capabilities. The modernized architecture addresses this challenge by providing updated technologies for use by the combatant commands and services to delivery joint training that is distributed where and when it is needed, tailored to respective missions, and providing relevant and realistic training content to challenge the force (SPAWAR Pacific, 2014).

The DODAF 2.0 approach is used to document the modernized joint training architecture. The effort includes the development of "as-is" and "to-be" architectures. Figure 14 shows the planned viewpoints for development of the architectures. As the "as-is" architecture evolved, the OV-5a, 5b models used business process modeling notation as this method provided the most comprehensive knowledge of the joint training enterprise. The "to-be" architecture is driving towards a workflow centric solution, which creates an integrated system-of -systems training environment.

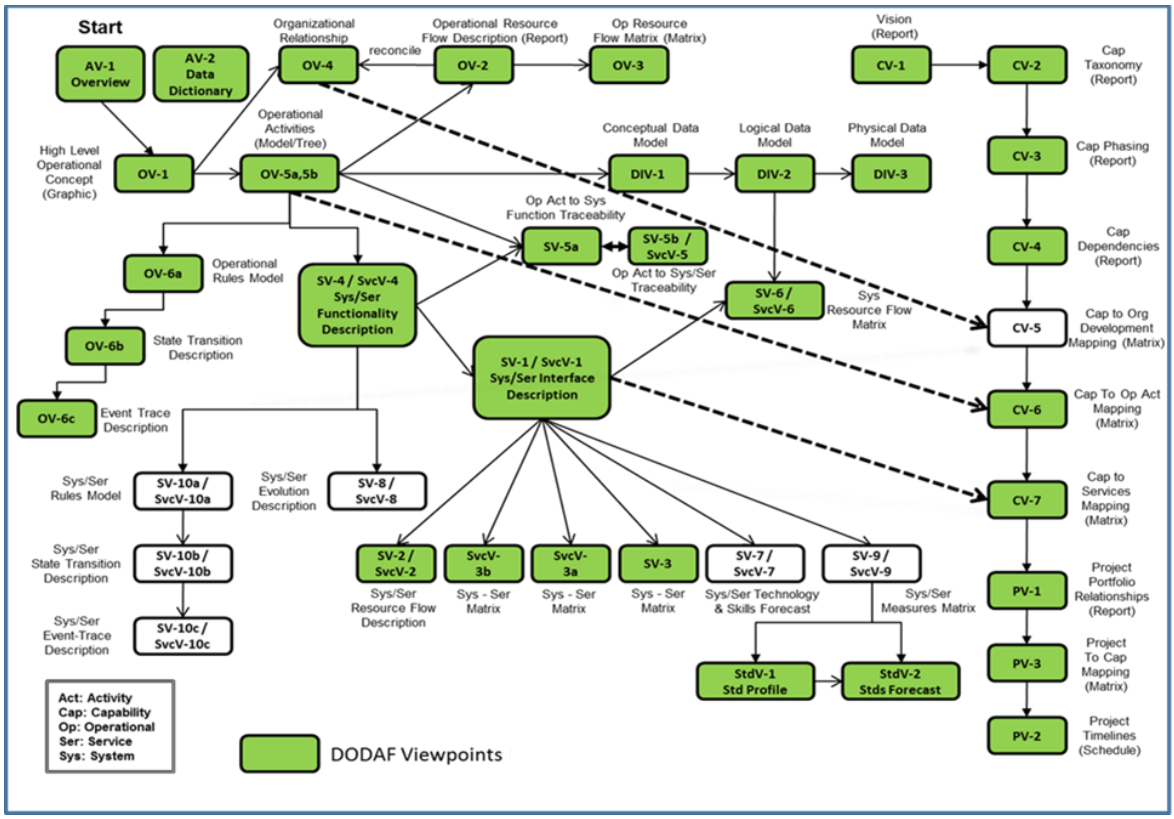

Figure 14. Joint training architecture viewpoints developed using DODAF 2.0 (from SPAWAR Pacific 2014a).

Section II.B, Overview of DODAF Viewpoints and Models, describes the typical representation used to develop each of the models. The following criteria are established in this research to determine the baseline set of DODAF models to consider for possible implementation using MP:

1. The DODAF model is graphically represented;
2. The model has the implementation of precedence relations;
3. The model has the implementation of inclusion relations.

It is not enough for the model to have only a graphical representation; it must also have precedence and/or inclusion to generate MP visualizations. Given these criteria, Table 8 determines the evaluation status of the DODAF viewpoints and models.

Table 9. Evaluation status of DODAF models for MP approach.

Viewpoints	MP Approach	
	Models Not Evaluated	**Models Evaluated**
All	AV-1, AV-2	None
Capability	CV-1,CV-3, CV-4, CV-5, CV-6, CV-7	CV-2
Data and Information	DIV-1, DIV-2, DIV-3	None
Operational	OV-1, OV-3, OV-6a	OV-2, OV-4, OV-5a, OV-5b, OV-6b, OV-6c
Project	PV-2, PV-3	PV-1
Services	SvcV-1, SvcV-3a, SvcV-3b, SvcV-5, SvcV-6, SvcV-7, SvcV-8, SvcV-9, SvcV-10a	SvcV-2, SvcV-4, SvcV-10b, SvcV-10c
Standards	StdV-1, StdV-2	None
Systems	SV-1, SV-3a, SV-3b, SV-5, SV-6, SV-7, SV-8, SV-9, SV-10a	SV-2, SV-4, SV-10b, SV-10c

B. DERIVED MP MODEL(S)

The following sections detail the conducted research including a description of the model under analysis, the method of conversion to MP, the MP code, MP visualizations, and a summary of results. The models evaluated are based on those identified in Table 8. Since model representations within each DODAF viewpoint are repeated, only one use case per model type is analyzed.

1. Capability Taxonomy: CV-2

The CV-2 is a hierarchical diagram used to develop taxonomy of capabilities for the architecture. Figure 15 shows a joint training model of capabilities. The model has six levels and 30 capabilities. For this research, the area outlined in red is modeled using MP.

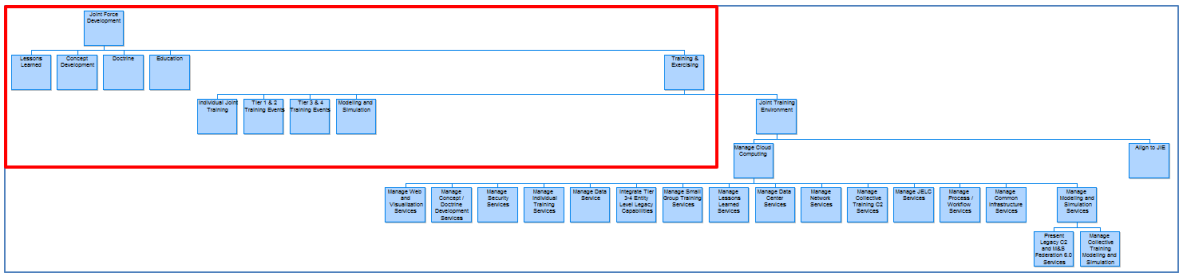

Figure 15. Joint training capability taxonomy model (DODAF CV-2) (from SPAWAR Pacific 2015).

a. Method of Conversion

Hierarchies are easy to code in MP and there are two approaches documented in this research. For the first approach, each level is defined as a ROOT with the capabilities defined as events of the root. The events must exist and no specific order is required. The MP notation for unordered events is A: {B,C, …Z}. This notation would exponentially grow the event traces, which is not a desired result for hierarchical diagrams. However, ordering does make a difference in the readability of the diagram and is suggested that the modeler pay attention to which events decompose at the next lower level in arrangement of events in the MP code. Next, the COORDINATE statement is used to link the events to the levels defined in the ROOT.

The second approach is much simpler and produces a similar hierarchical model. In this approach, a single ROOT event defines the top level of the hierarchy and the subsequent levels are nested as events within the single ROOT.

b. CV-2 MP Code

Below the two sets of code developed to model the first three levels and ten capabilities of the CV-2 use case are shown in Figure 15. Since the events must exist and no specific order is required, only one trace event is generated.

```
SCHEMA CV2_CapabilityTaxonomy_Version_One
/* ----------------------LEVELS------------------------ */
ROOT Capability_Taxonomy_Level_0:
                          joint_force_development;
ROOT Capability_Taxonomy_Level_1:
                          (lessons_learned
                                        concept_development
                          doctrine
                          education
                          training_and_exercising);
ROOT Capability_Taxonomy_Level_2:
                          (individual_joint_training
                          tier_1_2_training_events
                          tier_3_4_training_events
                          modeling_and_simulation);
/* --------------------INTERACTIONS--------------------- */
COORDINATE
     $x: joint_force_development   FROM
     Capability_Taxonomy_Level_0,
     $y: lessons_learned      FROM
     Capability_Taxonomy_Level_1
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: joint_force_development   FROM
     Capability_Taxonomy_Level_0,
     $y: concept_development  FROM
     Capability_Taxonomy_Level_1
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: joint_force_development   FROM
     Capability_Taxonomy_Level_0,
     $y: doctrine             FROM
     Capability_Taxonomy_Level_1
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: joint_force_development   FROM
     Capability_Taxonomy_Level_0,
     $y: education            FROM
     Capability_Taxonomy_Level_1
```

```
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: joint_force_development   FROM
     Capability_Taxonomy_Level_0,
     $y: training_and_exercising   FROM
     Capability_Taxonomy_Level_1
          DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: training_and_exercising   FROM
     Capability_Taxonomy_Level_1,
     $y: individual_joint_training FROM
     Capability_Taxonomy_Level_2
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: training_and_exercising   FROM
     Capability_Taxonomy_Level_1,
     $y: tier_1_2_training_events  FROM
     Capability_Taxonomy_Level_2
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: training_and_exercising   FROM
     Capability_Taxonomy_Level_1,
     $y: tier_3_4_training_events  FROM
     Capability_Taxonomy_Level_2
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: training_and_exercising   FROM
     Capability_Taxonomy_Level_1,
     $y: modeling_and_simulation   FROM
     Capability_Taxonomy_Level_2
          DO ADD $x PRECEDES $y; OD;
```

Second version of CV-2 MP code:

```
SCHEMA CV2_Capability_Taxonomy_Version_Two
/* ------------------------------------------------------
                    ACTORS
------------------------------------------------------ */
ROOT joint_force_development:
                    lessons_learned
                    concept_development
                    doctrine
                    education
                    training_and_exercising;
training_and_exercising: individual_joint_training
                    tier_1_2_training_events
```

```
tier_3_4_training_events
```

c. CV-2 MP Visualizations

Note that the events are linked in the order defined in both sets of code, which is not necessary for a purely hierarchical diagram. For the version one code, the MP Analyzer generates three visualizations shown prior to any manual manipulation in Figure 16. The simpler, nested code (version two) initial and manipulated visualizations are shown in Figure 18 and Figure 19.

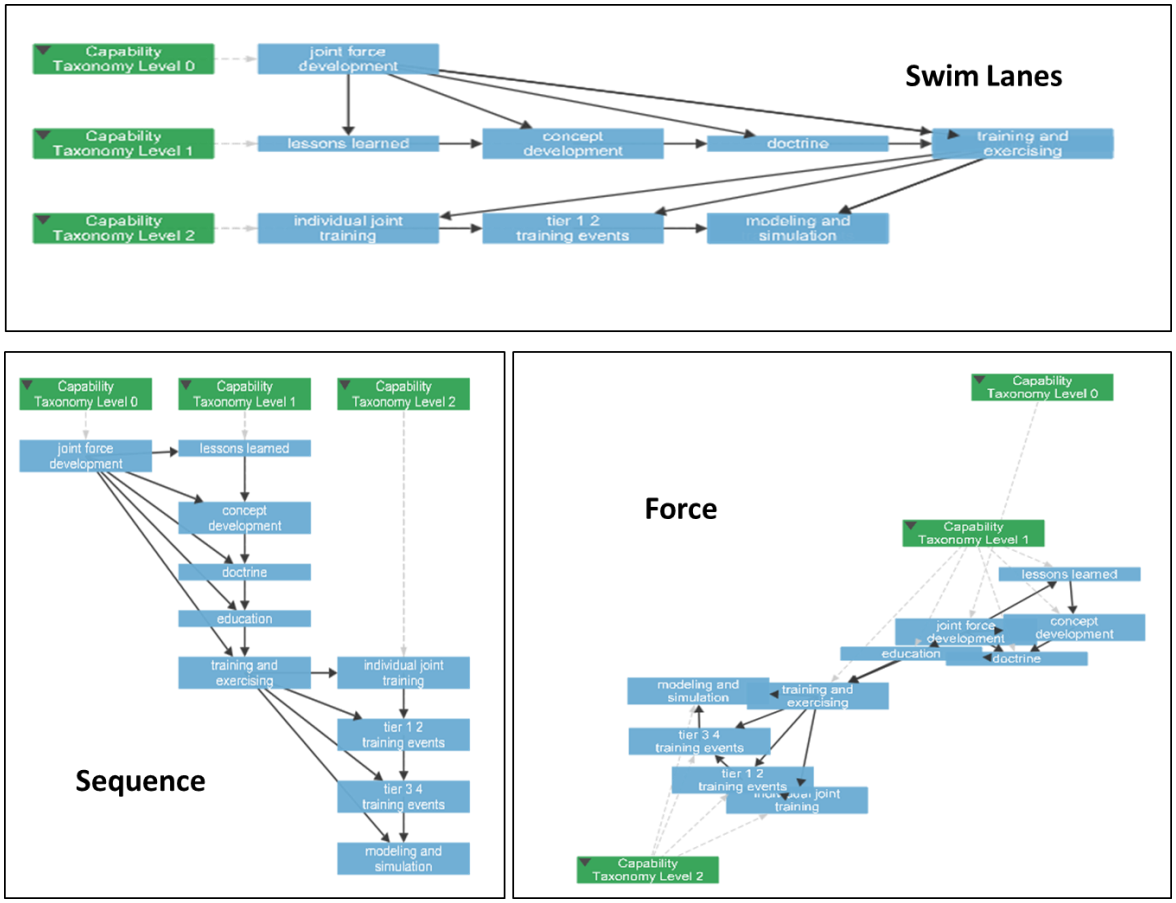

Figure 16. MP Analyzer swim lanes, sequence, and force visualizations for CV-2, MP code version one.

The resulting MP Analyzer generated swim lane is most easily manipulated for a comparable DODAF CV-2 visualization as shown in Figure 17.

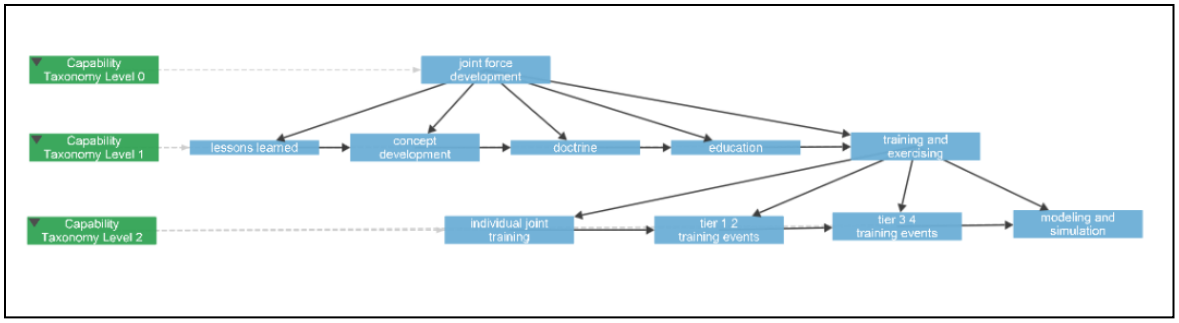

Figure 17. Manipulated MP CV-2 swim lane visualization, MP code version one.

The CV-2 MP code version two generates the following results shown in Figure 18.

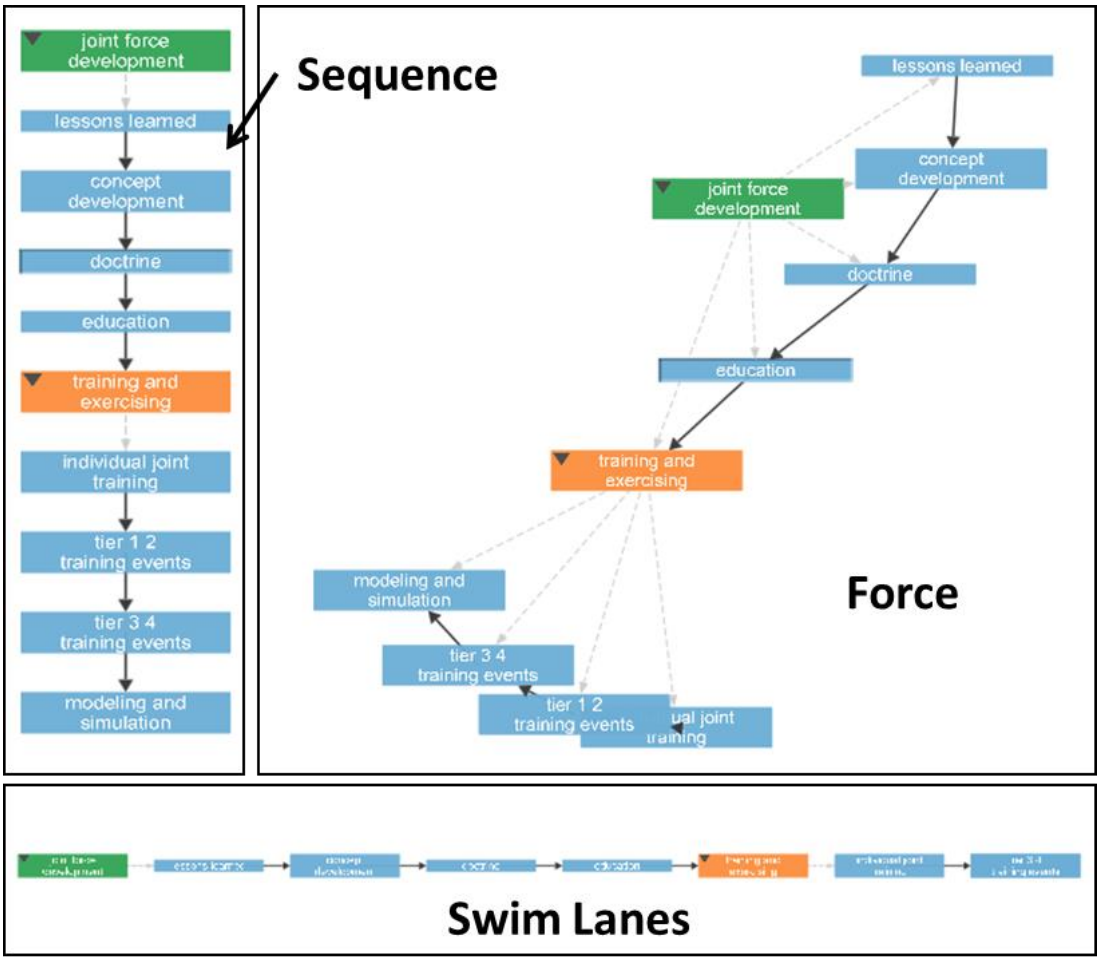

Figure 18. MP Analyzer swim lanes, sequence and force visualizations for CV-2, MP code version two.

Each of the above visualizations is easily manipulated for a comparable DODAF CV-2 visualization as shown in Figure 19.

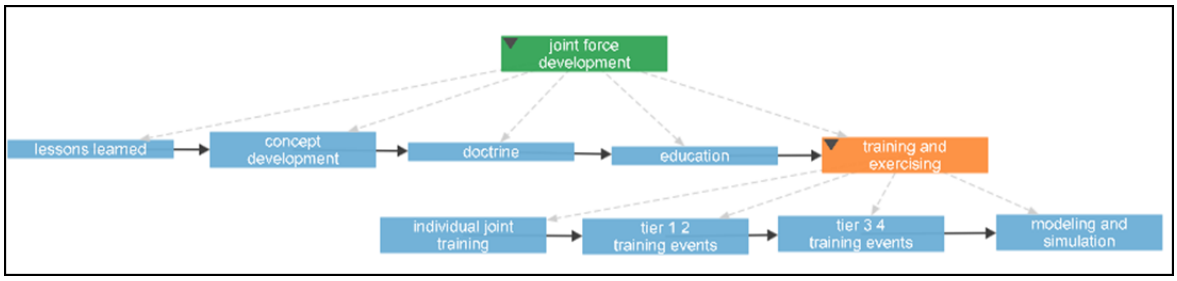

Figure 19. Manipulated MP CV-2, code version two.

d. CV-2 MP Summary

While this model is easily generated using MP, there are no scenarios that warrant the analysis of multiple event traces as only the levels of the hierarchy are relevant; the order of the capabilities within the level typically are not of concern to a modeler. Additionally, only one of the MP visualizations represents the intent of the model well and requires manual intervention for optimal display to include exposing the "education" event, which is initially hidden under the "doctrine" event.

2. Resource Flow Diagrams: OV-2, SvcV-2, and SV-2

The OV-2, SvcV-2, and SV-2 are resource flow diagrams, which show the flow of resources from one entity to another. To illustrate the conversion of these DODAF models to MP, analysis is conducted only on a subset of the SV-2 in Figure 5. To simplify the model and illustrate the conversion to MP, the yellow highlighted area of the SV-2 is re-created in Figure 20.

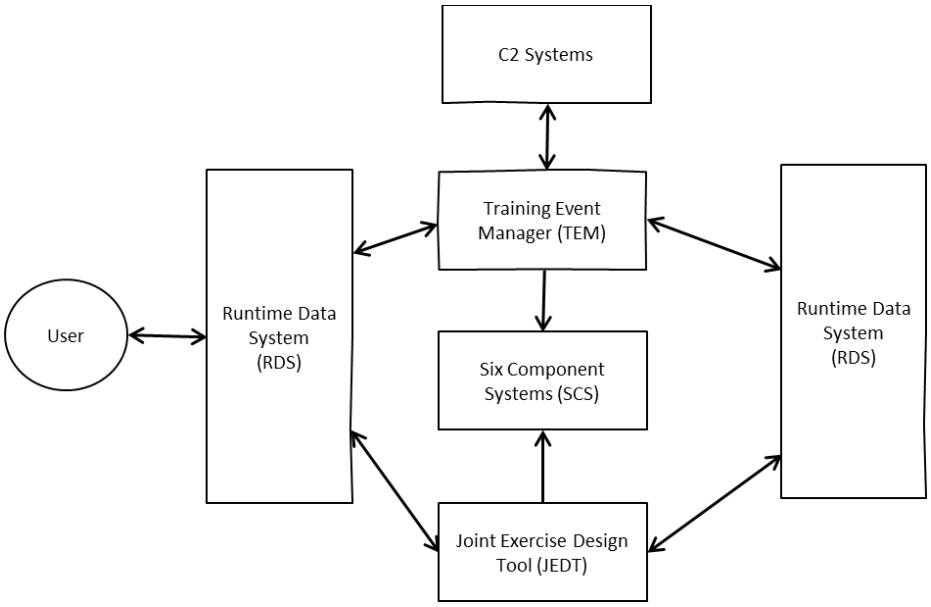

Figure 20. Subset of joint training SV-2.

The traditional DODAF model shows resource flows to/from entities; however, the model does not communicate some simple architectural concerns such as precedence. Is the precedence of the resource flows important? For example, does the Training Event Manager (TEM) have to receive something from the C2 Systems prior to being able to send resources to the Runtime Data System (RDS)? What behaviors are associated with the resource flows from system to system? Additionally, as previously discussed, the SV-2 shows that the Six Component Systems (SCS) receive resources from the TEM and Joint Exercise Design Tool (JEDT), but does not output any resources to any other system in the SV-2.

a. Method of Conversion

The first step in converting the SV-2 is to make some assumptions about the resource relationships between the systems in the model. Since the connectors are bi-directional, the assumption is made that the systems have behaviors (processes) to get and send data between each other. The second step is to model the processes within each system required for the flow of resources. The next step is to build the MP code by defining each system using the ROOT statement, each process as the events in the ROOT, and defining the interactions of the systems using the COORDINATE statement.

Modeling specific system behaviors is a key principle of the MP approach, particularly when modeling a system of systems. The updated SV-2, ready for MP code implementation is shown in Figure 21.

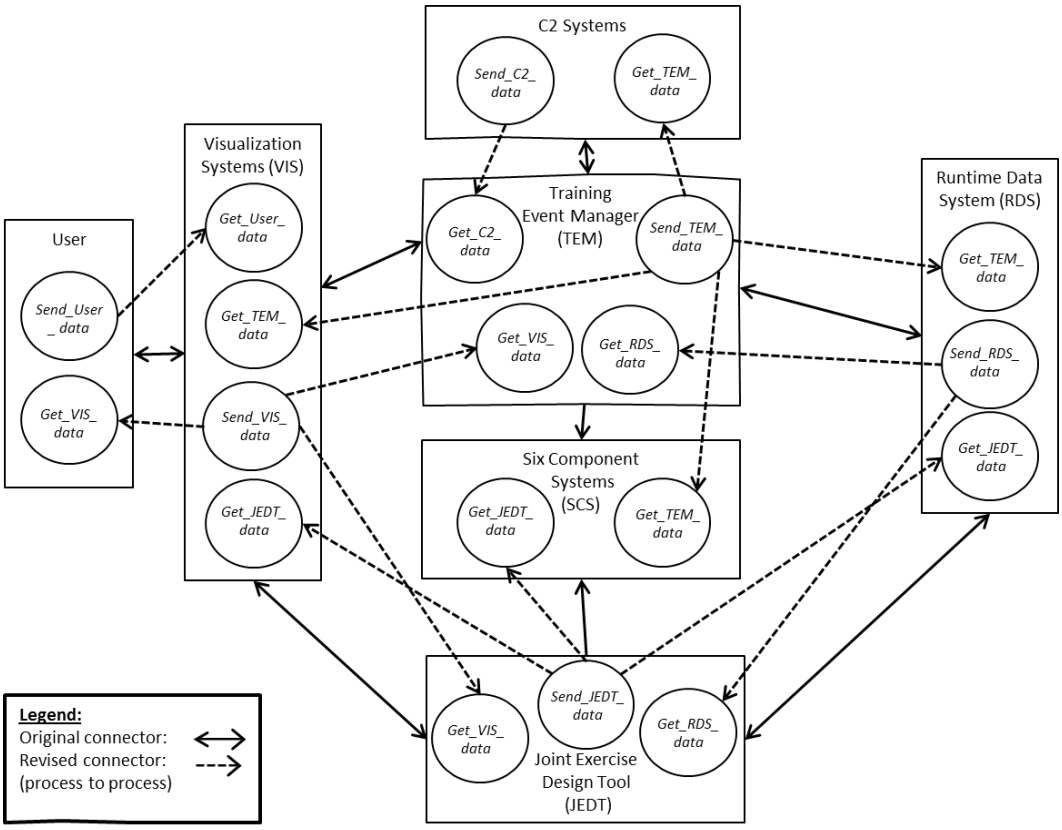

Figure 21. Revised SV-2 model showing the system behaviors.

b. SV-2 MP Code

Below is the code developed to model the revised SV-2 use case shown in Figure 5. The model has seven systems, 20 events, and 14 interactions. The order and precedence of the events are unknown in this SV-2 and presumed to be set as coded; therefore, only one trace event is generated.

```
SCHEMA SystemsView
/* ------------------------------------------------------
                    Systems
------------------------------------------------------ */
ROOT C2_Systems:
                          send_C2_data
```

```
                        get_TEM_data;
ROOT TEM:
                        get_C2_data
                        send_TEM_data
                        get_vis_data
                        get_RDS_data;
ROOT Comp_Systems:
                        get_TEM_data
                        get_JEDT_data;
ROOT JEDT:
                        send_JEDT_data
                        get_vis_data
                        get_RDS_data;
ROOT RDS:
                        send_RDS_data
                        get_TEM_data
                        get_JEDT_data;
ROOT Vis_Systems:
                        send_vis_data
                        get_TEM_data
                        get_JEDT_data
                        get_User_data;
ROOT User:
                        send_User_data
                        get_vis_data;
/* ---------------------------------------------------------
                    Resource Flows
--------------------------------------------------------- */
COORDINATE     $x: send_C2_data         FROM C2_Systems,
               $y: get_C2_data          FROM TEM
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_TEM_data        FROM TEM,
               $y: get_TEM_data         FROM C2_Systems
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_TEM_data        FROM TEM,
               $y: get_TEM_data         FROM Vis_Systems
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_JEDT_data       FROM JEDT,
               $y: get_JEDT_data        FROM Vis_Systems
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_TEM_data        FROM TEM,
               $y: get_TEM_data         FROM Comp_Systems
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_JEDT_data       FROM JEDT,
               $y: get_JEDT_data        FROM Comp_Systems
               DO ADD $x PRECEDES $y; OD;
```

```
COORDINATE     $x: send_vis_data        FROM Vis_Systems,
               $y: get_vis_data         FROM User
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_User_data       FROM User,
               $y: get_User_data        FROM Vis_Systems
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_TEM_data        FROM TEM,
               $y: get_TEM_data         FROM RDS
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_RDS_data        FROM RDS,
               $y: get_RDS_data         FROM TEM
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_JEDT_data       FROM JEDT,
               $y: get_JEDT_data        FROM RDS
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_RDS_data        FROM RDS,
               $y: get_RDS_data         FROM JEDT
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_vis_data        FROM Vis_Systems,
               $y: get_vis_data         FROM JEDT
               DO ADD $x PRECEDES $y; OD;
COORDINATE     $x: send_vis_data        FROM Vis_Systems,
               $y: get_vis_data         FROM TEM
               DO ADD $x PRECEDES $y; OD;
```

c. SV-2 MP Visualizations

Figure 22 and Figure 23 show the three MP visualizations generated from MP Analyzer.

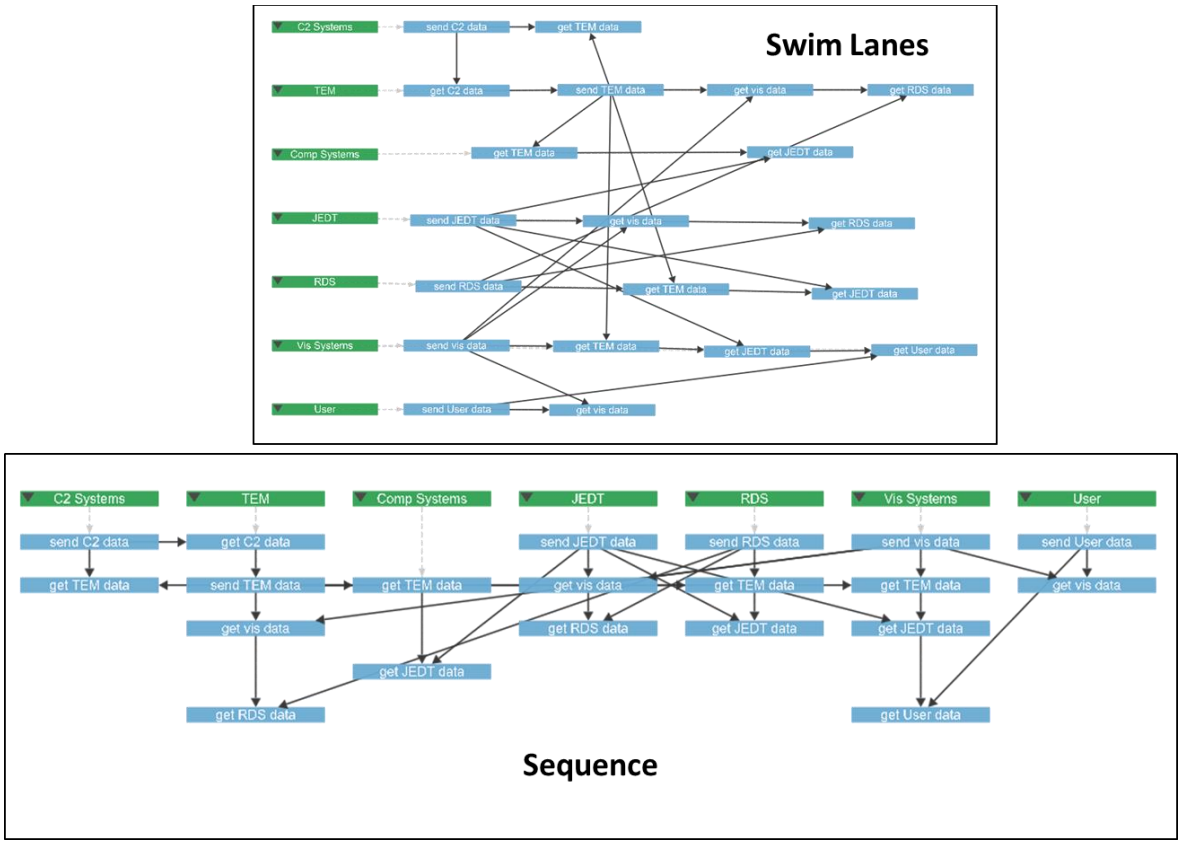

Figure 22. MP swim lanes and sequence visualizations generated from SV-2.

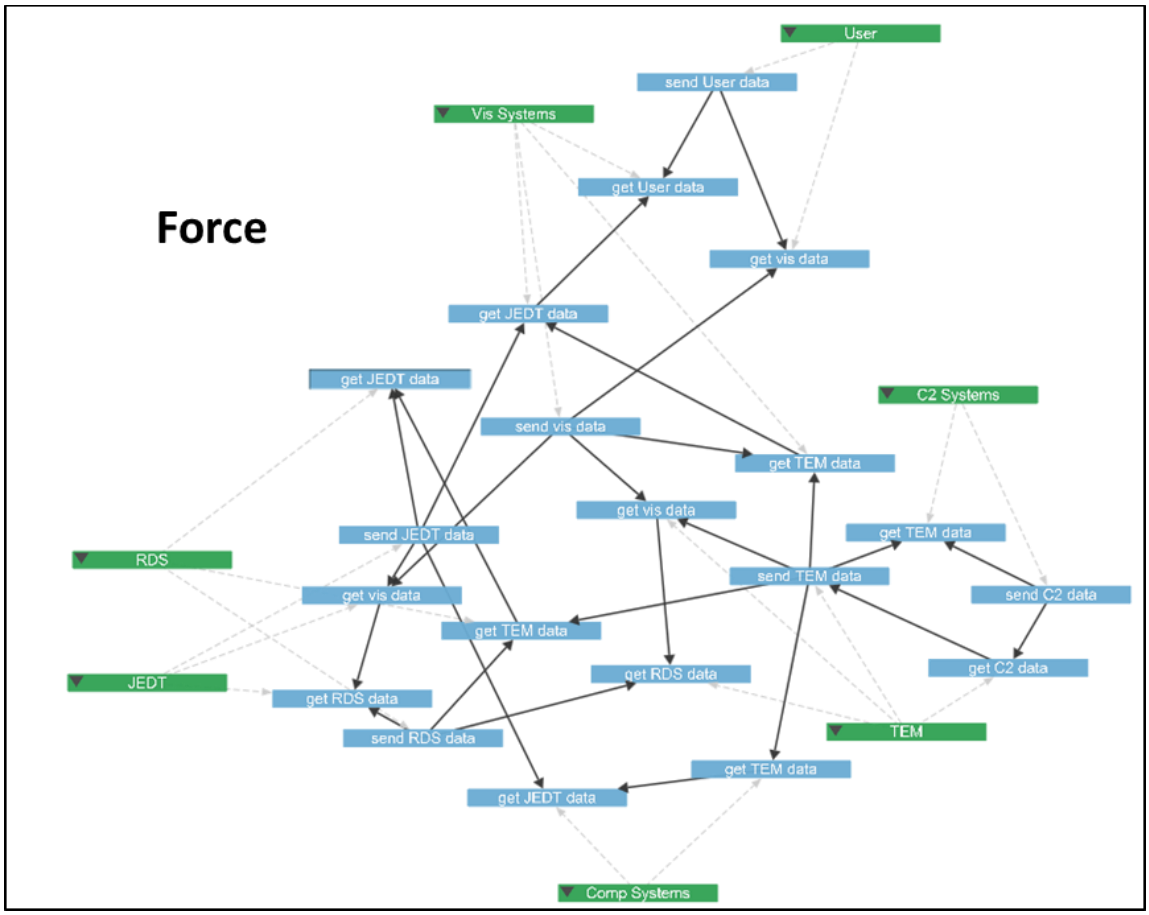

Figure 23. MP force visualization generated from SV-2.

d. SV-2 (OV-2 and SvcV-2) MP Summary

Resource flow diagrams become complex very quickly and in the use case model presented here, the logical correctness of this diagram is questionable. Using the MP approach, the modeler can easily implement the important concerns of the model – the behaviors between the systems, services, or operational activities. The MP models provide excellent representation the DODAF SV-2 use case. DODAF OV-2 and SvcV-2 models can also be similarly modeled. An OV-2 emerges from the SV-2 by simply collapsing the ROOT events in the SV-2 model as seen in Figure 24. Using MP, a single code instance generates both models.

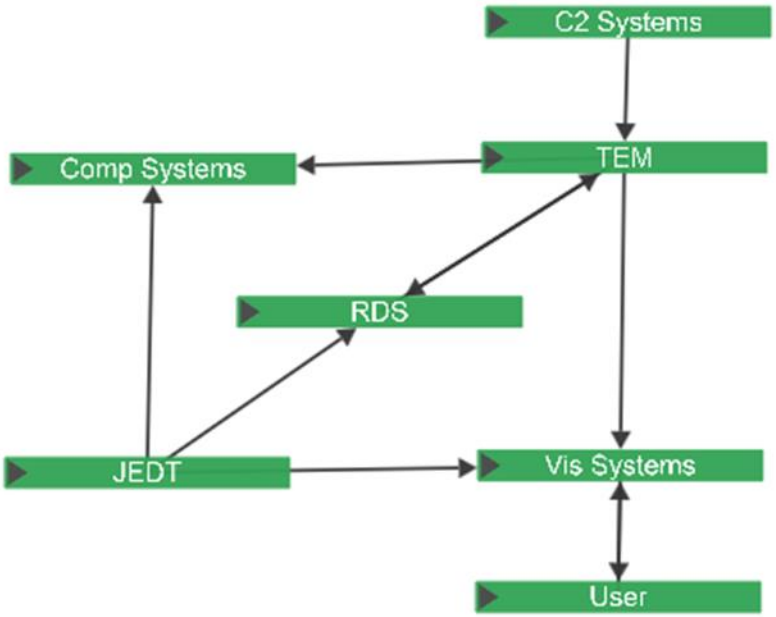

Figure 24. OV-2 generated from collapse of SV-2 MP ROOT events.

As modeled for this research, only one event trace is generated based on the known information. However, the systems architect must pursue the answers to the questions about how the resources actually are intended to flow. Once those answers are known, multiple event traces emerge.

3. Organizational and Project Relationship Charts: OV-4 and PV-1

Organizational and project relationship charts are hierarchies and can be modeled in the same approach used for the CV-2 with the same methods and results. For purposes of this research, only a sample OV-4 is modeled as proof of DODAF to MP generation. For the OV-4 model, additional analysis is conducted to depict the organizational "assistant" construct and the usage of the model to communicate not just the role itself, but quite often, a named individual. An example model, rather than an actual one, was developed and is shown in Figure 25.

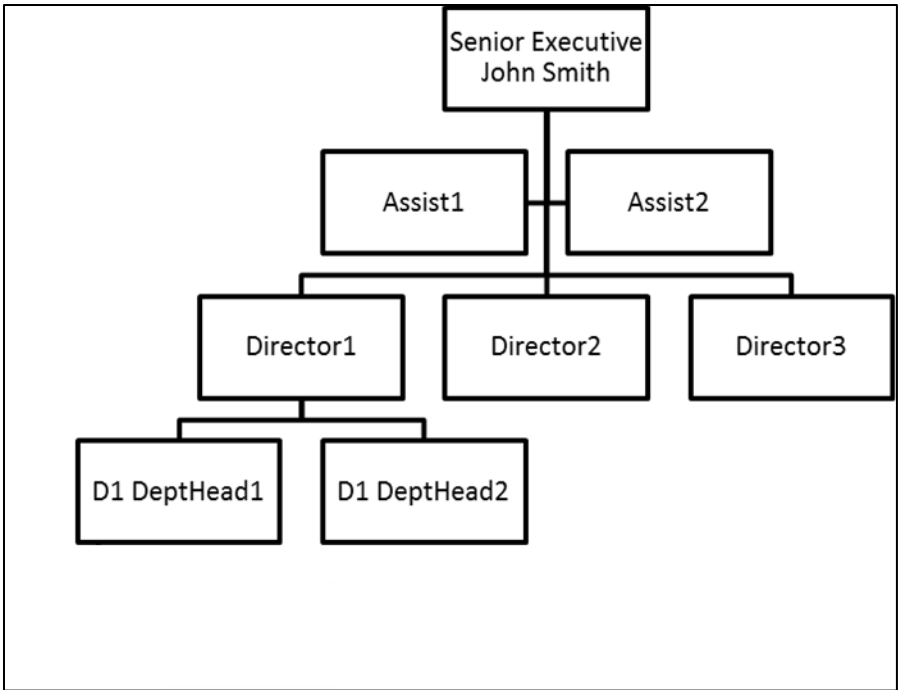

Figure 25. Example OV-4, organizational relationship chart.

a. Method of Conversion

The methods of conversion are identical to those used for the CV-2.

b. OV-4 MP Code

Below is the code developed to model the first four levels and eight organizational elements of the OV-4 use case shown in Figure 25.

```
SCHEMA OV4_OrgChart
/* ------------------------------------------------------
                    Organizational Levels
------------------------------------------------------ */
ROOT Org_Level_0:
               SeniorExecutive_John_Smith;
ROOT Org_Level_1_Assists:
               (Assist1
               Assist2);
ROOT Org_Level_2:
               (Director1
               Director2
               Director3);
ROOT Org_Level_3:
```

```
               (D1_DeptHead1
               D1_DeptHead2);
/* ------------------------------------------------------
                    Interactions
------------------------------------------------------ */
COORDINATE
          $x:SeniorExecutive_John_Smith FROM Org_Level_0,
          $y: Assist1 FROM    Org_Level_1_Assists
          DO ADD $x PRECEDES $y; OD;
COORDINATE
          $x:SeniorExecutive_John_Smith FROM Org_Level_0,
          $y: Assist2 FROM    Org_Level_1_Assists
          DO ADD $x PRECEDES $y; OD;
COORDINATE
          $x:SeniorExecutive_John_Smith FROM Org_Level_0,
          $y: Director1  FROM Org_Level_2
          DO ADD $x PRECEDES $y; OD;
COORDINATE
          $x:SeniorExecutive_John_Smith FROM Org_Level_0,
          $y: Director2  FROM Org_Level_2
          DO ADD $x PRECEDES $y; OD;
COORDINATE
          $x:SeniorExecutive_John_Smith FROM Org_Level_0,
          $y: Director3  FROM Org_Level_2
          DO ADD $x PRECEDES $y; OD;
COORDINATE
          $x: Director1       FROM Org_Level_2,
          $y: D1_DeptHead1    FROM Org_Level_3
          DO ADD $x PRECEDES $y; OD;
COORDINATE
          $x: Director1       FROM Org_Level_2,
          $y: D1_DeptHead2    FROM Org_Level_3
          DO ADD $x PRECEDES $y; OD;
```

c. OV-4 MP Visualizations

The MP model in Figure 26 shows that “assistants” modeled as their own level; however, the visual distinction of their role is not as clearly defined in the MP model when compared to a traditional organizational chart.

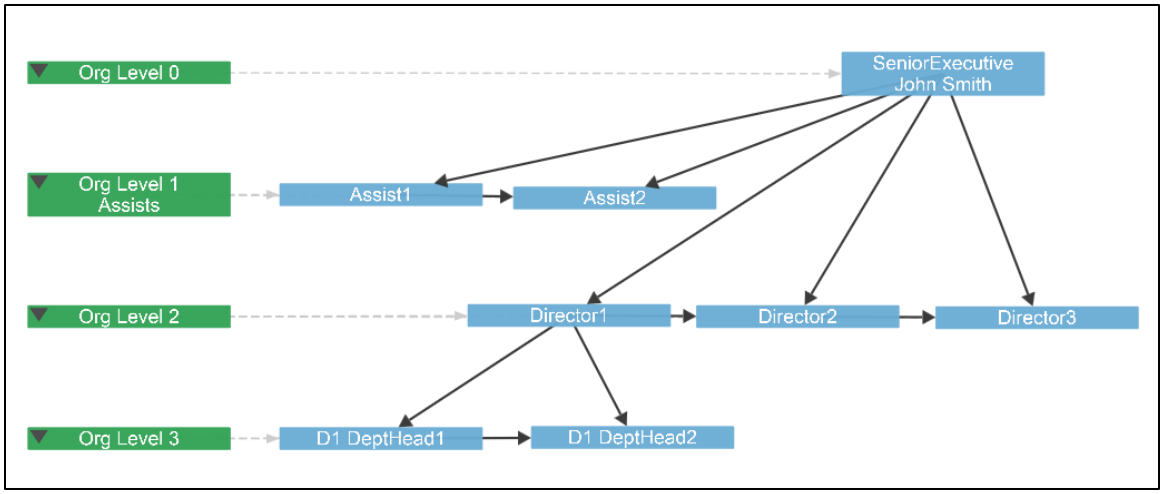

Figure 26. Manipulated MP Analyzer swim lanes visualization for OV-4.

d. OV-4 and PV-1 MP Summary

The results mirror the ones discussed for the CV-2. Other modeling tools provide more robust and flexible capabilities for developing organizational and project charts. However, the data required to generate these DODAF models in MP exists. Additionally, the PV-1 and PV-2 are tightly coupled as the PV-2 is used to model the project timelines, tasks, and key milestones and is typically modeled using project scheduling tools and GANTT charts.

4. Operational Activity Models: OV-5a and OV-5b

An OV-5a is an operational activity decomposition model, which is represented as a hierarchy and is used to reference the OV-5b. In the use case shown below, the activity "prepare for the exercise" is decomposed from the OV-5a. The generation of hierarchical models in MP is shown through the use case research for the DODAF CV-2 and OV-4 models. The OV-5b is an operational activity model diagram that shows relationships among activities including inputs and outputs. Figure 27 shows the operational activities for a joint training use case model for the activity "prepare for the exercise." The OV-5b has four activities, eleven originating inputs, and eight destination outputs. For this research, the activities connected by the inputs/outputs highlighted in red are modeled in MP.

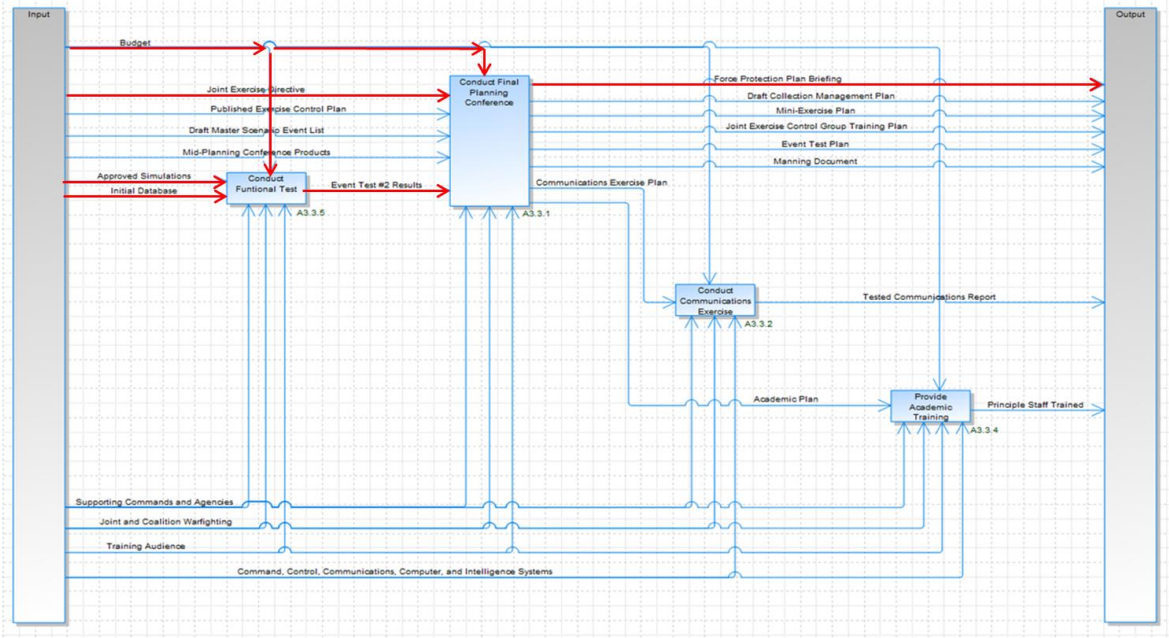

Figure 27. Joint training activity model (OV-5b), prepare for the exercise (from SPAWAR Pacific 2013).

a. Method of conversion

Activity modeling in MP is straightforward. First, activities from the OV-5b are represented using the ROOT statement. Next, the data flows of each activity are described as events of the ROOT. The COORDINATE statement is used to model the interactions between the ROOT activities. The OV-5b activities are "Input," "Conduct Functional Test," "Conduct Final Planning Conference," and "Output." The OV-5b inputs/outputs are modeled as events of the activities as listed in the ROOT statements of the OV-5b MP code.

b. OV-5b MP Code

```
SCHEMA PrepareExercise
/* ------------------------------------------------------
                          Activities
------------------------------------------------------ */
ROOT Input:
                    send_budget
                    send_initial_database
                    send_approved_simulations;
ROOT conduct_functional_test:
```

```
                    get_budget
                    get_approved_simulations
                    get_initial_database
                    send_event_test2_results;
ROOT conduct_final_planning_conference:
                    get_budget
                    get_joint_exercise_directive
                    get_event_test2_results
                    send_force_protection_plan_briefing;
ROOT Output:
                    get_force_protection_plan_briefing;
/* ------------------------------------------------------
                          INTERACTIONS
------------------------------------------------------ */
COORDINATE
     $x:  send_approved_simulations     FROM Input,
     $y:  get_approved_simulations      FROM
          conduct_functional_test
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:  send_initial_database    FROM Input,
     $y:  get_initial_database FROM conduct_functional_test
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:  send_budget         FROM Input,
     $y:  get_budget          FROM conduct_functional_test
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:  send_event_test2_results FROM
          conduct_functional_test,
     $y:  get_event_test2_results  FROM
          conduct_final_planning_conference
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:  send_budget    FROM Input,
     $y:  get_budget     FROM
          conduct_final_planning_conference
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:  send_force_protection_plan_briefing     FROM
          conduct_final_planning_ conference,
     $y:  get_force_protection_plan_briefing FROM Output
     DO ADD $x PRECEDES $y; OD;
```

c. OV-5b MP Visualizations

Figure 28 and Figure 29 show the three MP visualizations generated from MP Analyzer.

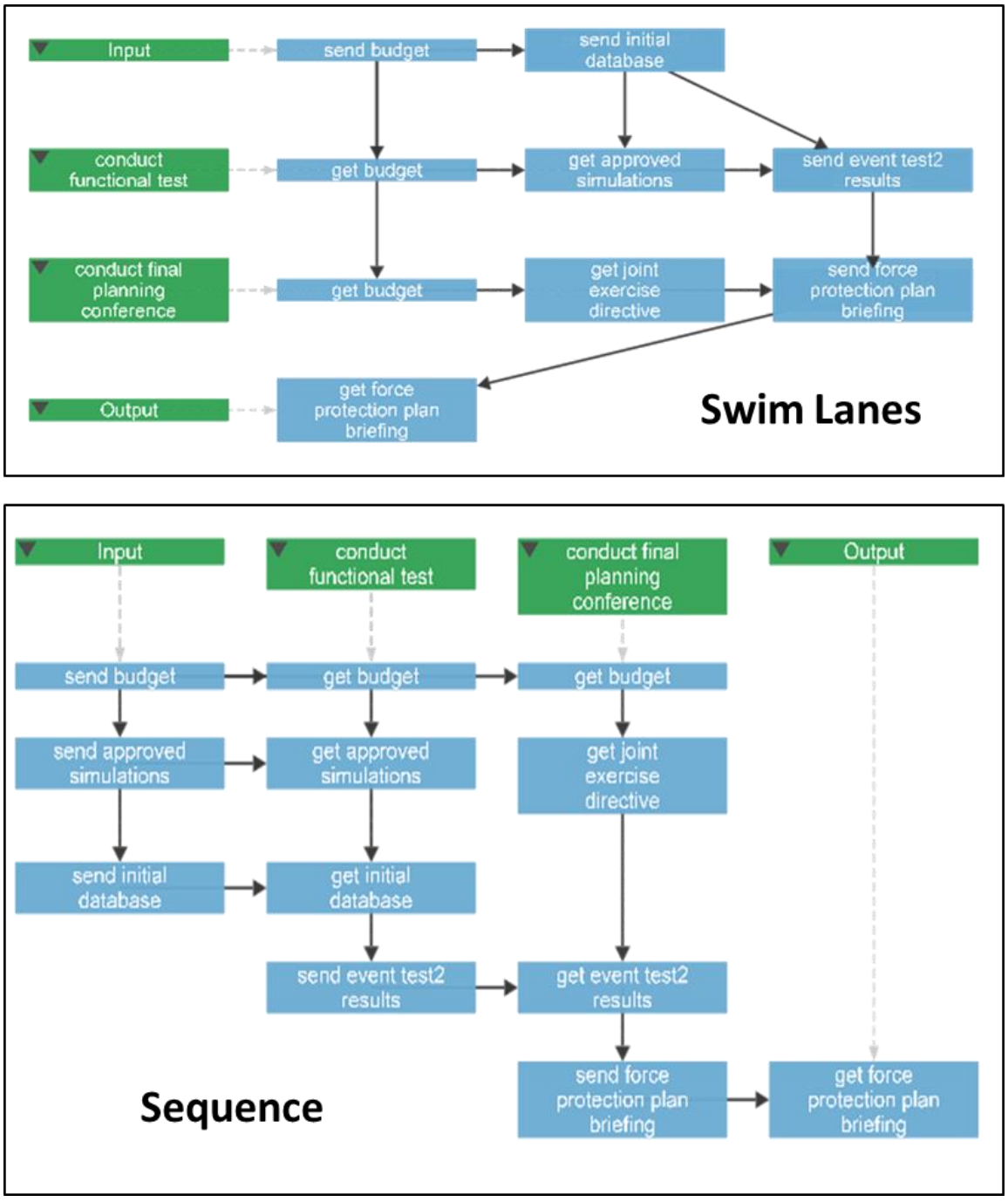

Figure 28. MP swim lanes and sequence visualizations for OV-5b.

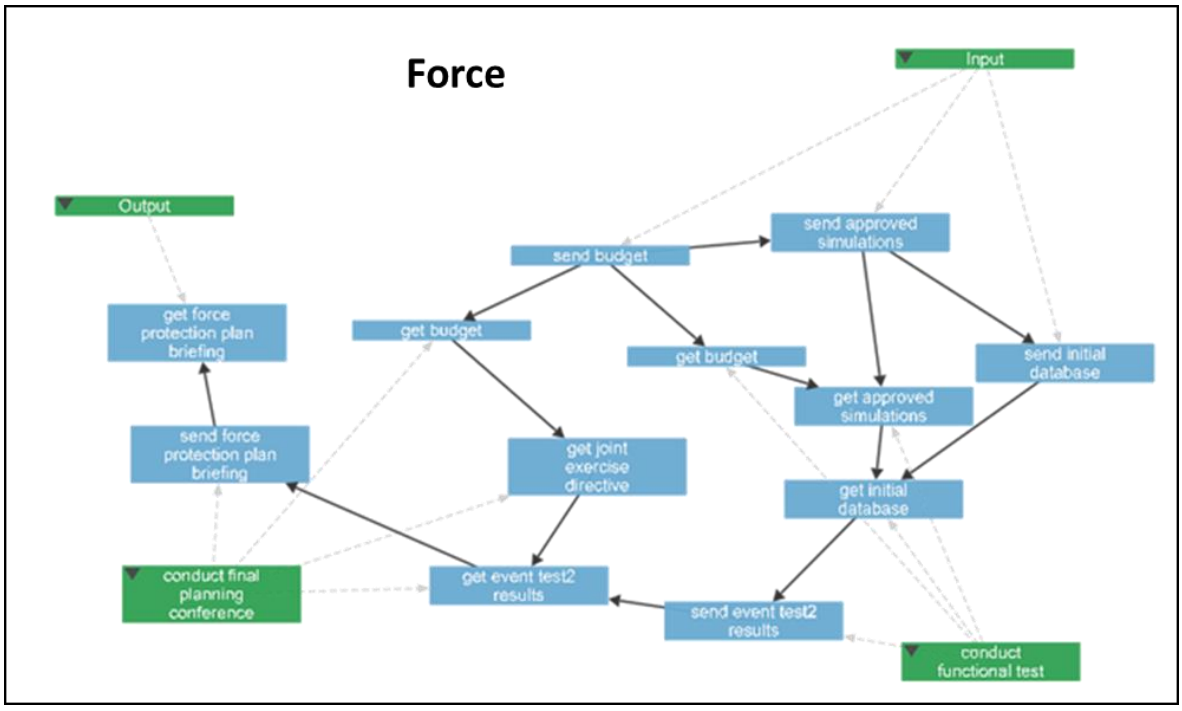

Figure 29. MP force visualization for OV-5b.

d. OV-5b MP Summary

Since behaviors are core to the MP approach, coding activities models in MP is a natural fit for realizing MP benefits. The MP swim lane and sequence visualizations are excellent representations of the example OV-5b and are easy to read and interpret. As modeled for this research, only one event trace is generated. However, since the "Input" and "Output" activities are actually abstractions for source and destination actors in the OV-5b, additional modeling work can be performed that would provide further resolution of the architecture and additional discovery benefits using MP.

5. State Transition Description: OV-6b, SvcV-10b, and SV-10b

The DODAF OV-6b, SvcV-10b, and SV-10b models are represented graphically as state transition diagrams. State transition diagrams are useful in describing the behavior of a single object by identifying all of its possible states. State transition diagrams have the characteristics to be modeled using MP: events, precedence and inclusion. Figure 30 shows an OV-6b for order processing states.

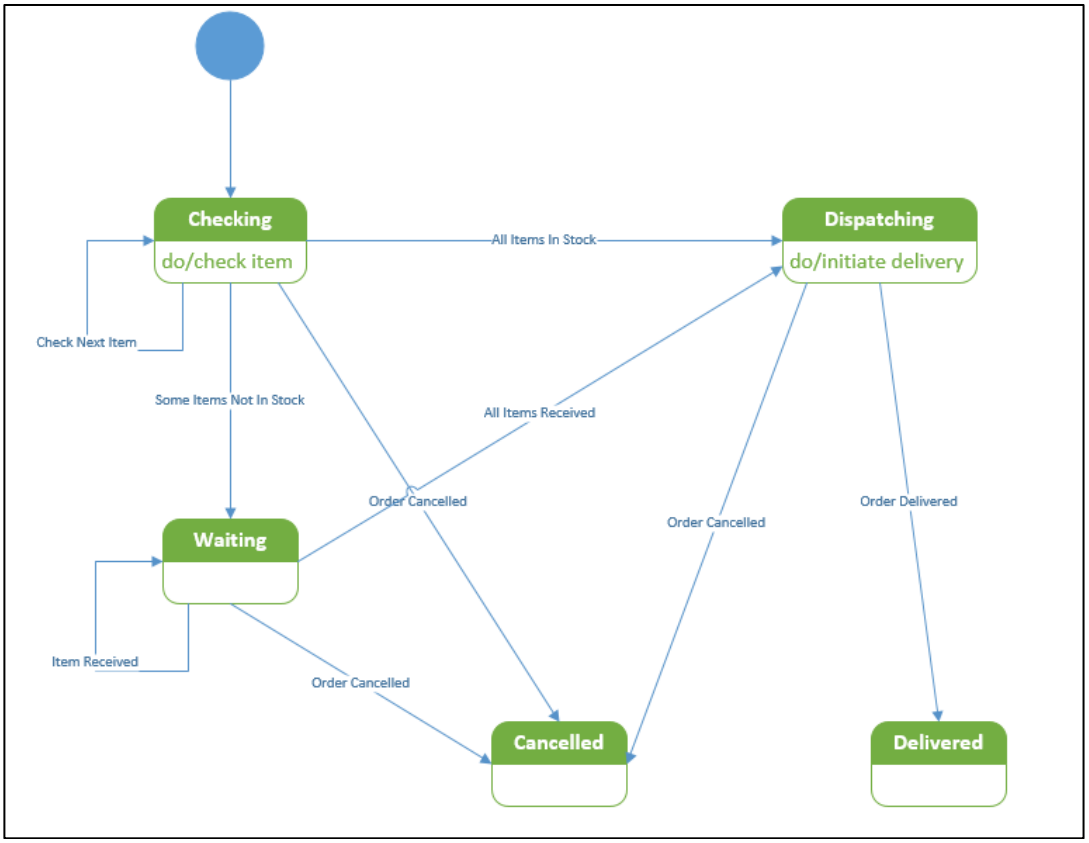

Figure 30. Order processing state diagram (after Fowler and Scott 1997).

a. Method of Conversion

The state diagram is modeled using one ROOT event to represent the order and a set of nested events to represent the states and transitions of the order as it is processed. The order states (checking, waiting, dispatching, cancelled, and delivered) are modeled and tested first. Next, the state transitions are added to the MP code to incrementally test the model event trace results. This iterative approach quickly exposed errors with the model as seen in Figure 31. This event trace reveals an undesirable end state of "waiting" and the "waiting, item_received, waiting*"* states occurring after the "cancelled"state. At scope two, 36 event traces are generated. At scope three, 60 event traces are generated.

Figure 31. Undesirable end state "waiting" discovered in MP event trace.

b. SvcV-10b MP Code

```
SCHEMA StateDiagram
/*-------------------------------------------------------
    Order Processing State Transition
    Starts at checking order
    Ends at order delivered or order cancelled
-------------------------------------------------------*/
ROOT OrderProcessing: (
                        /* Checking State */
                        Checking           (*Check_Next_Item
Checking*)
```

```
                        (
                          (
                            /* Dispatching State */
                            All_Items_In_Stock Dispatching
                            (
                              Order_Cancelled Cancelled |
                              Order_Delivered Delivered
                            )
                          ) |
                          (
                            /* Waiting State */
                            Some_Items_Not_In_Stock  Waiting
(*Item_Received Waiting*)
                            (
                              /* Dispatching State */
                              All_Items_Received
Dispatching
                              (
                                Order_Cancelled Cancelled |
                                Order_Delivered Delivered
                              ) |
                                Order_Cancelled Cancelled
                            )
                          ) |
                          (
                            Order_Cancelled Cancelled
                          )
                        )
                      );
```

c. SvcV-10b Visualizations

The MP Analyzer sequence visualization provides the best results for the state transition diagram. At scope one, 18 trace events are generated, at scope two, 36 trace events are generated, and at scope three, 60 trace events are generated. Each event trace ends with one of the acceptable end state results: cancelled or delivered. Four of the event trace sequence visualizations are shown in Figure 32.

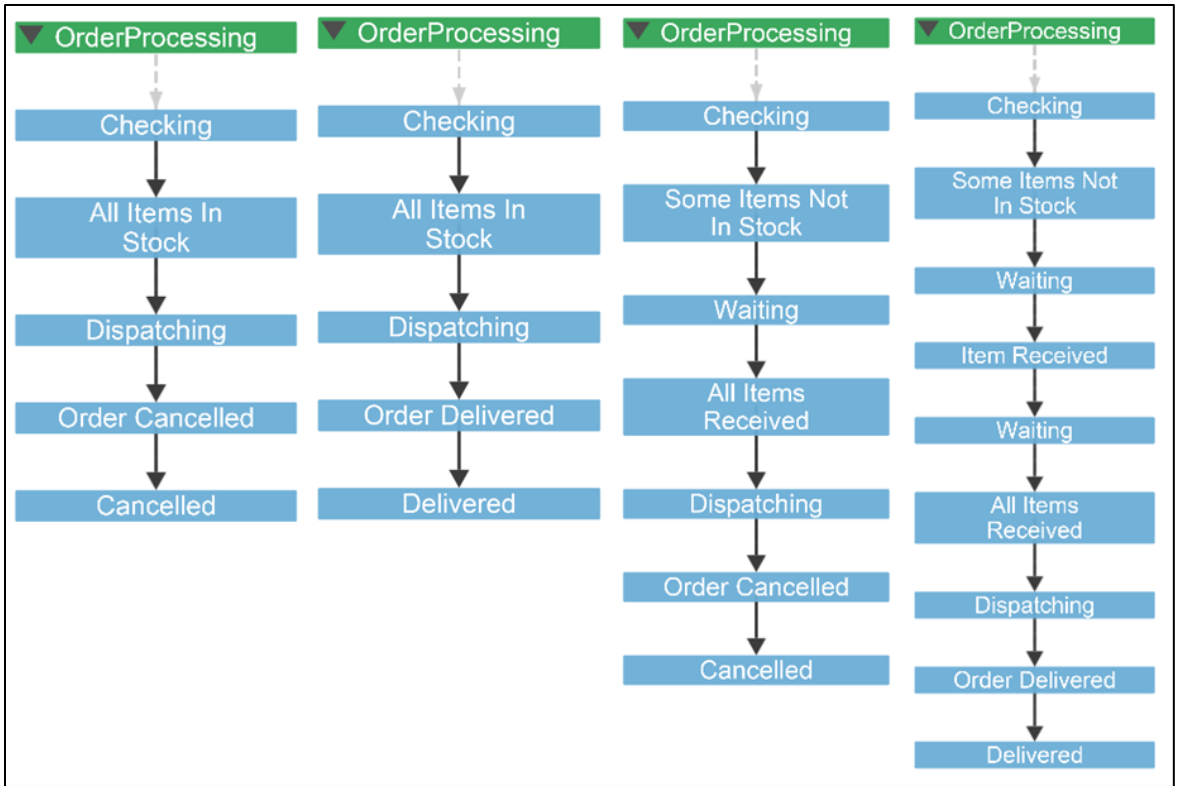

Figure 32. Four of 60 event trace sequence visualizations generated in MP Analyzer for OV-6b.

d. SvcV-10b Summary

The state transition diagram is modeled using only MP nested events and a single ROOT event to represent the order itself and events to represent the states and the transitions between the states. The MP Analyzer sequence diagram displays the event traces without any required manipulations. The transition loops to the state events are implemented in MP code using the (*A*) zero or more events pattern.

6. Operational, Services and Systems Event-Trace Descriptions: OV-6c, SvcV-10c, and SV-10c

The OV-6c, SvcV-10c, and SV-10c models are all event trace models. Each of these models is used to define functionality and sequences of events for operational, service, or system views. An initial OV-6c use case was developed to explore the research possibilities for this thesis and the model and corresponding code, visualizations are available in Appendix B. Baseline DODAF OV-6c . This baseline model generated

only one event trace at scopes one, two, and three when run in MP, which was cause for concern about the design. Further analysis of the process model revealed gaps in the design. This model was revised to complete missing design requirements such as such as parallel processes for notification, process loops for approvals, sub-processes, and decision gates for mission execution. The OV-6c use case in Figure 33 uses BPMN 2.0 to model the operational activities in the response mission training process. The design gap revisions are highlighted in yellow.

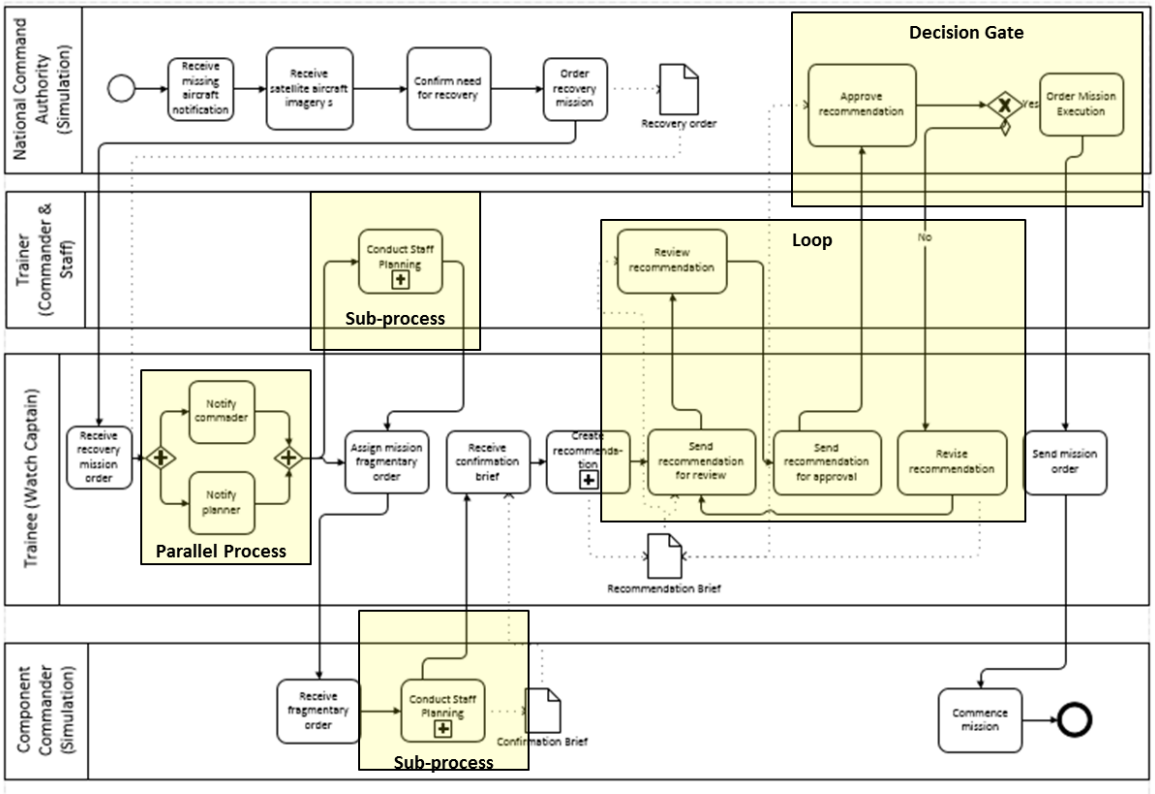

Figure 33. Response mission training thread (after SPAWAR Pacific 2014b).

a. Method of Conversion

BPMN 2.0 provides a robust set of constructs to model processes. BPMN 2.0 models can generate business process execution language (BPEL) which can be used in BPMN engines to simulate or execute the process model. The first step in converting this model is to map the BPMN 2.0 constructs to the MP approach as shown in Table 9.

Table 10. BPMN 2.0 definitions and mapping to MP (OMG 2011).

BPMN 2.0	BPMN 2.0 Definition	MP
Activity (Task, Transaction, Event Sub-Process, Call Activity)	Work that a company or organization performs using business processes. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a process model are process, sub-process, and task.	Events (inclusion and/or precedence)
Data Object	The primary construct for modeling data within the process.	Events (inclusion and/or precedence)
End Event	An event that indicates where a path in the process will end.	Events (inclusion and/or precedence)
Flow (Sequence, Default, Conditional)	A directional connector between elements in a process.	Interaction between events (precedence)
Gateways	A construct used to route the sequence flow of events in the process: parallel or decision based events	Events (inclusion and/or precedence)
Message Flow	A connecting object that shows the flow of messages between two participants.	Interactions (inclusion and/or precedence)
Process	A sequence or flow of activities in an organization with the objective of carrying out work. In BPMN, a process is depicted as a graph of flow elements, which are a set of activities, events, gateways, and sequence flow that adhere to finite execution semantics.	Series of events (inclusion and/or precedence)
Start Event	An event that indicates where a particular process starts.	Events (inclusion and/or precedence)
Swim lane or Pool	A swim lane (or lane) is a graphical container for partitioning a set of activities from other activities. BPMN has two different types of swim lanes. Pool or lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.	Root events (inclusion)

The four swim lanes, National Command Authority, Trainer, Trainee, and Component Commander were defined as ROOT events. Next, the events in each swim lane are included in the ROOT events and the parallel events, loops, and decision gates are coded using the MP grammar. Finally, the interactions between the swim lanes are coded as interactions using the COORDINATE statement.

The parallel events, shown in Figure 34, with no ordering requirement, (Notify Commander and Notify Planner) within the ROOT event for the Trainee are easily coded using the MP grammar: {*notify_commander, notify_planner*}.

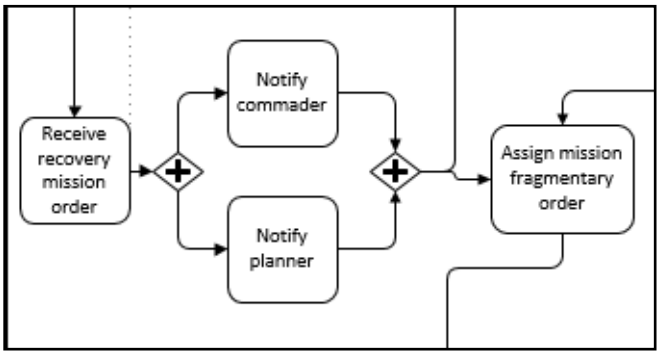

Figure 34. BPMN parallel gateways.

However, an additional event, *complete_notifications* is added to communicate that the parallel events finish prior to the start of the next events, *assign_mission_fragmentary_order and* the coordinated event, *conduct_staff_planning* as shown in Figure 35. The BPMN parallel gateway is modeled as an MP event.

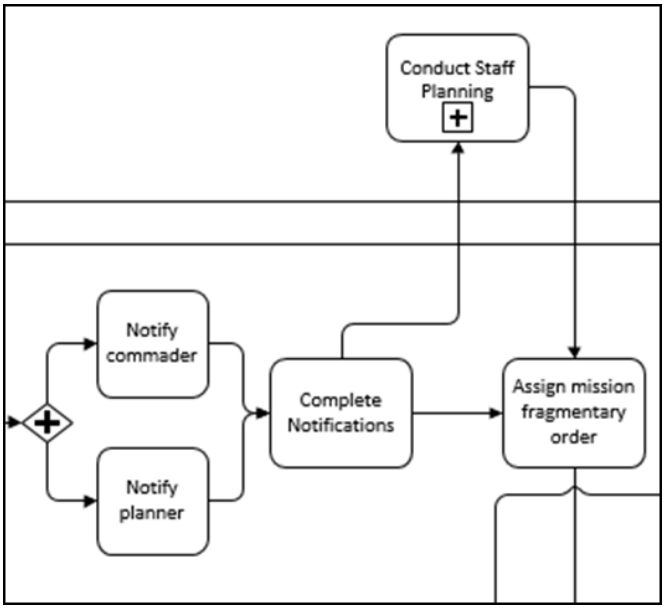

Figure 35. *Complete_notifications* activity enforces completion of the parallel tasks.

The MP Analyzer tool requires manual manipulation to separate the parallel events in the visualization as shown in the before and after in Figure 36.

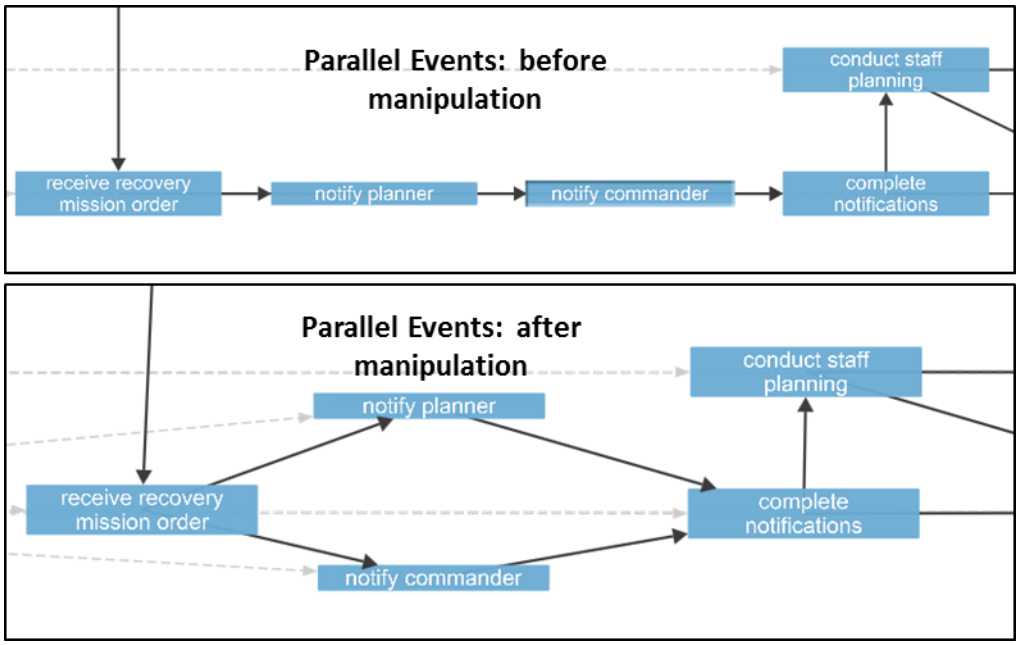

Figure 36. MP visualizations for parallel events.

The decision gateway also requires additional MP events in order to model the paths of the decision as interactions between the ROOT events and the conversion was not as simply made from the BPMN model to the MP code. Loop events, shown in Figure 37, occurring one or more time are modeled using the using the ordered sequence of event pattern, A: (+B+) and the COORDINATE statement between the ROOT events.

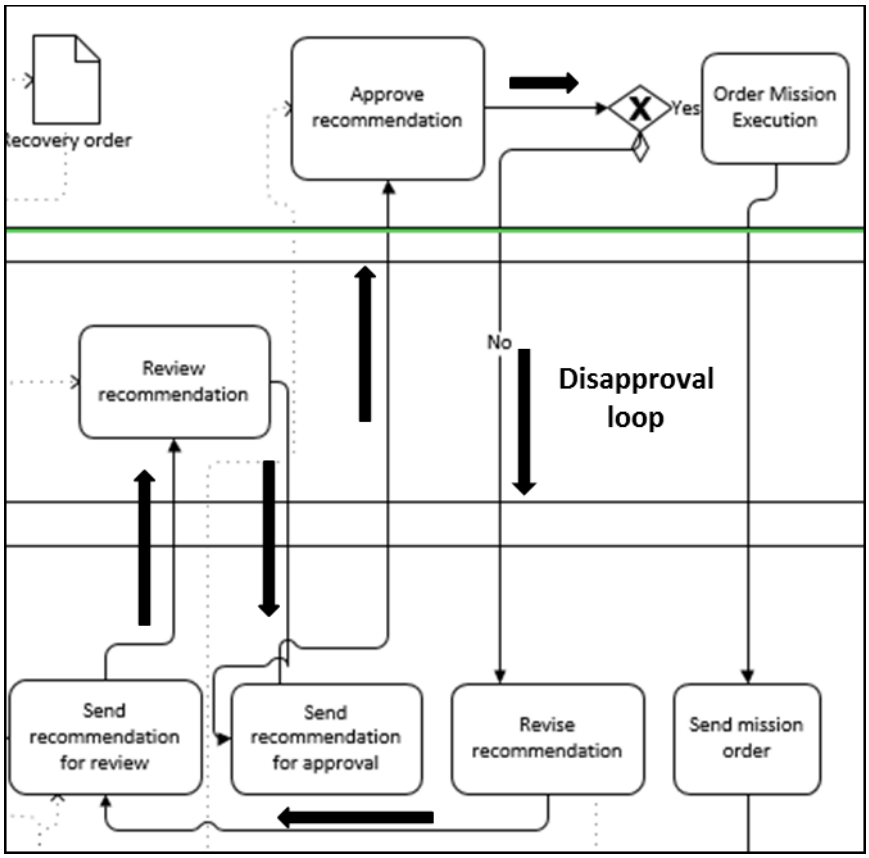

Figure 37. Loop event with decision gateway.

In order to model the BPMN decision gate in MP, events are created to represent the approval and disapproval activities highlighted in Figure 37.

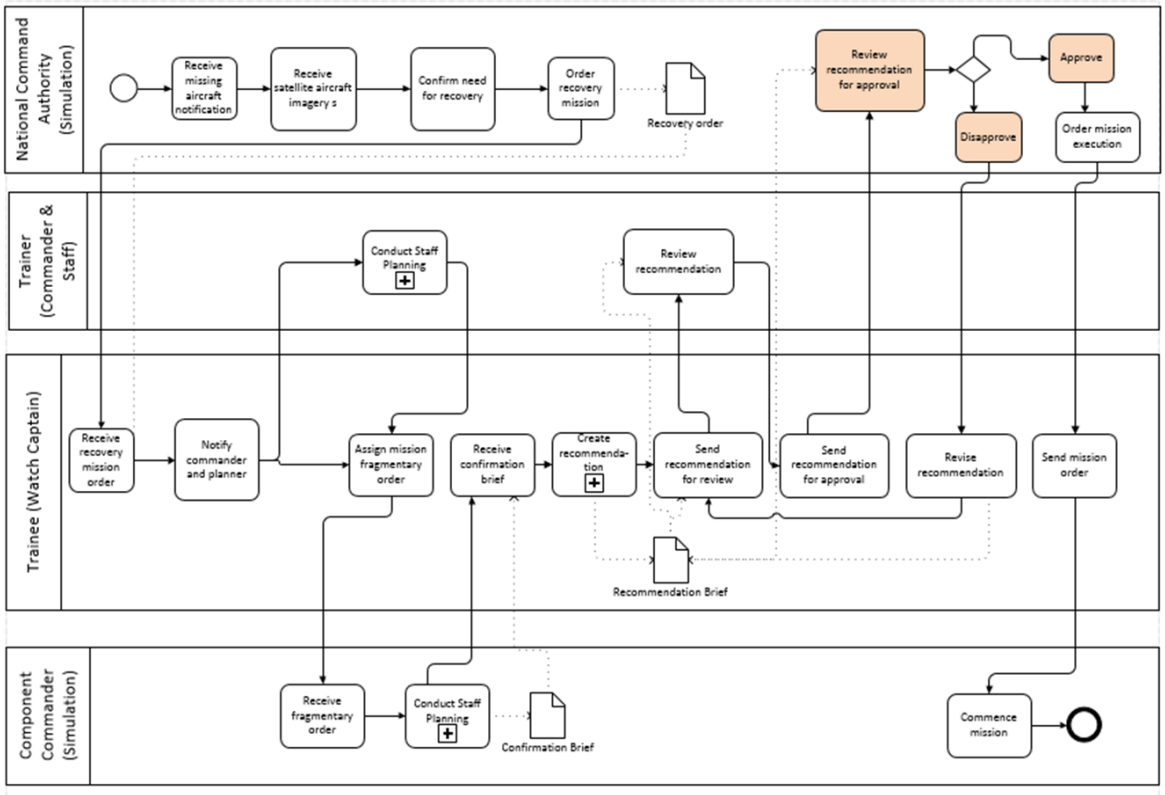

Figure 38. Modified BPMN model with disapprove and approve activities.

b. OV-6c MP Code – Final

Below is the final code developed to model the modified OV-6c. Since the order of the parallel events is not relevant for event tracing, the parallel events were combined for simplicity of the model.

```
SCHEMA ResponseMissionTraining
/* ------------------------------------------------
                    ACTORS
------------------------------------------------ */
ROOT National_Command_Authority_Sim:
     start
     receive_missing_aircraft_notification
     receive_satellite_aircraft_imagery
     confirm_recovery_need
     order_recovery_mission
     (+ review_recommendation_for_approval
     (approve order_mission_execution |
                                  disapprove) +) ;
ROOT Trainer_Commander_and_Staff:
```

```
     conduct_staff_planning
     (+ review_recommendation +) ;
ROOT Trainee_Watch_Captain:
     receive_recovery_mission_order
     notify_commander_and_planner
     complete_notifications
     assign_mission_fragmentary_order
     receive_confirmation_brief
     create_recommendation
     (+  send_recommendation_for_review
     send_recommendation_for_approval
     (revise_recommendation  |
     decides_cancel_the_mission |
     send_mission_order) +) ;
ROOT      Component_Commander_Sim:
     receive_fragmentary_order
     conduct_staff_planning
     (+ (commence_mission | abort_mission) +)
     end ;
/* ---------------------------------------------------
                    INTERACTIONS
--------------------------------------------------- */
COORDINATE
     $x: order_recovery_mission    FROM
     National_Command_Authority_Sim,
     $y: receive_recovery_mission_order      FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:complete_notifications     FROM
     Trainee_Watch_Captain,
     $y: conduct_staff_planning    FROM
     Trainer_Commander_and_Staff
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: conduct_staff_planning    FROM
     Trainer_Commander_and_Staff,
     $y:assign_mission_fragmentary_order     FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:assign_mission_fragmentary_order     FROM
     Trainee_Watch_Captain,
     $y:receive_fragmentary_order            FROM
     Component_Commander_Sim
     DO ADD $x PRECEDES $y; OD;
```

```
COORDINATE
     $x: conduct_staff_planning         FROM
     Component_Commander_Sim,
     $y: receive_confirmation_brief     FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:send_recommendation_for_review  FROM
     Trainee_Watch_Captain,
     $y: review_recommendation          FROM
     Trainer_Commander_and_Staff
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: review_recommendation               FROM
     Trainer_Commander_and_Staff,
     $y:send_recommendation_for_approval     FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:send_recommendation_for_approval     FROM
     Trainee_Watch_Captain,
     $y: review_recommendation_for_approval  FROM
     National_Command_Authority_Sim
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: disapprove                 FROM
     National_Command_Authority_Sim,
     $y:(revise_recommendation                          |
     decides_cancel_the_mission)    FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: order_mission_execution    FROM
     National_Command_Authority_Sim,
     $y: send_mission_order         FROM
          Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: send_mission_order    FROM
     Trainee_Watch_Captain,
     $y: commence_mission      FROM
     Component_Commander_Sim
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: decides_cancel_the_mission     FROM
     Trainee_Watch_Captain,
```

```
$y: abort_mission   FROM Component_Commander_Sim
DO ADD $x PRECEDES $y; OD;
```

c. OV6-c MP Visualizations

The OV6-c is run in MP Analyzer at scope one, two, and three generating, two, twelve, and 56 event traces. Only the scope one event traces are shown. The MP Analyzer sequence diagrams for the approval and disapproval traces are shown in Figure 39 and Figure 40.

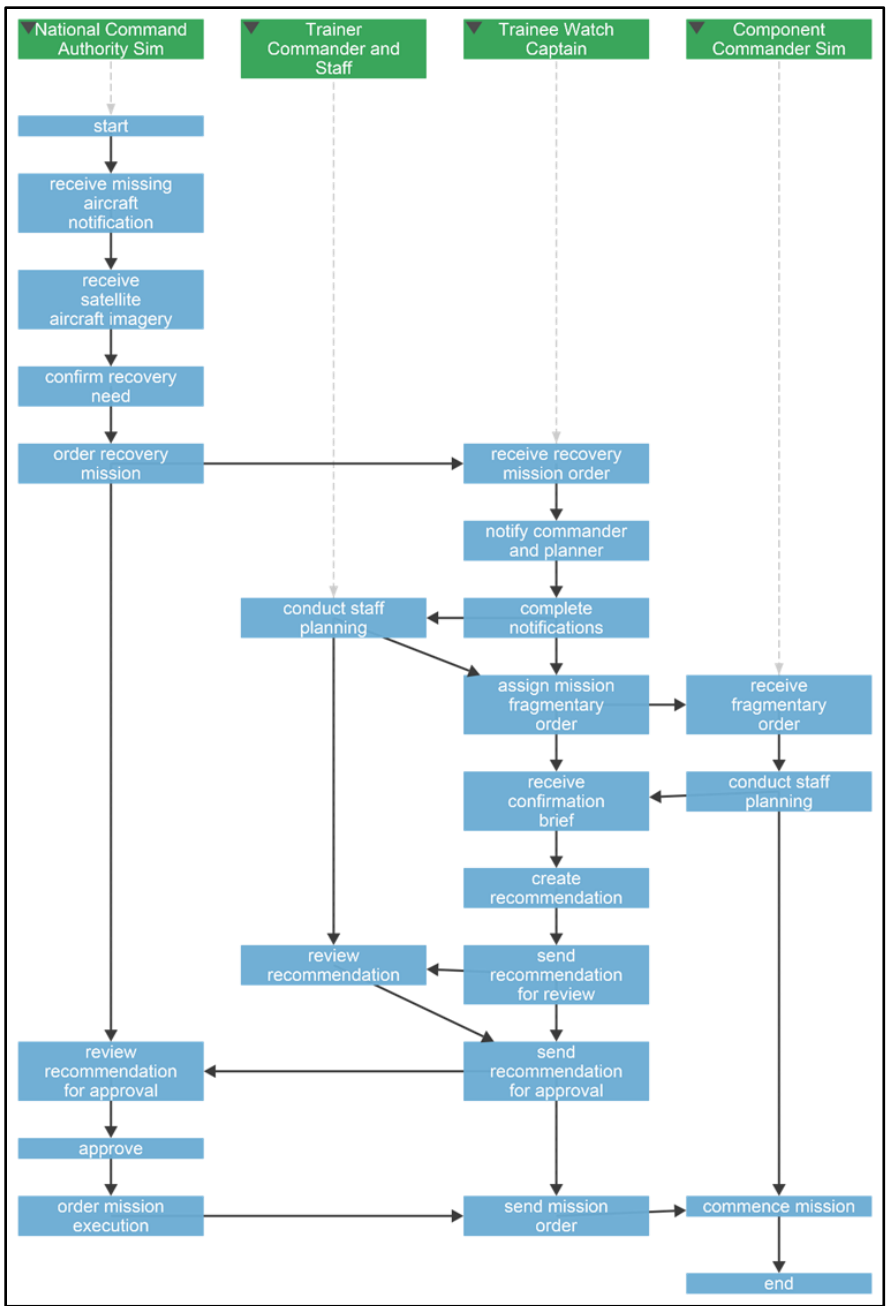

Figure 39. MP sequence diagram for approval decision.

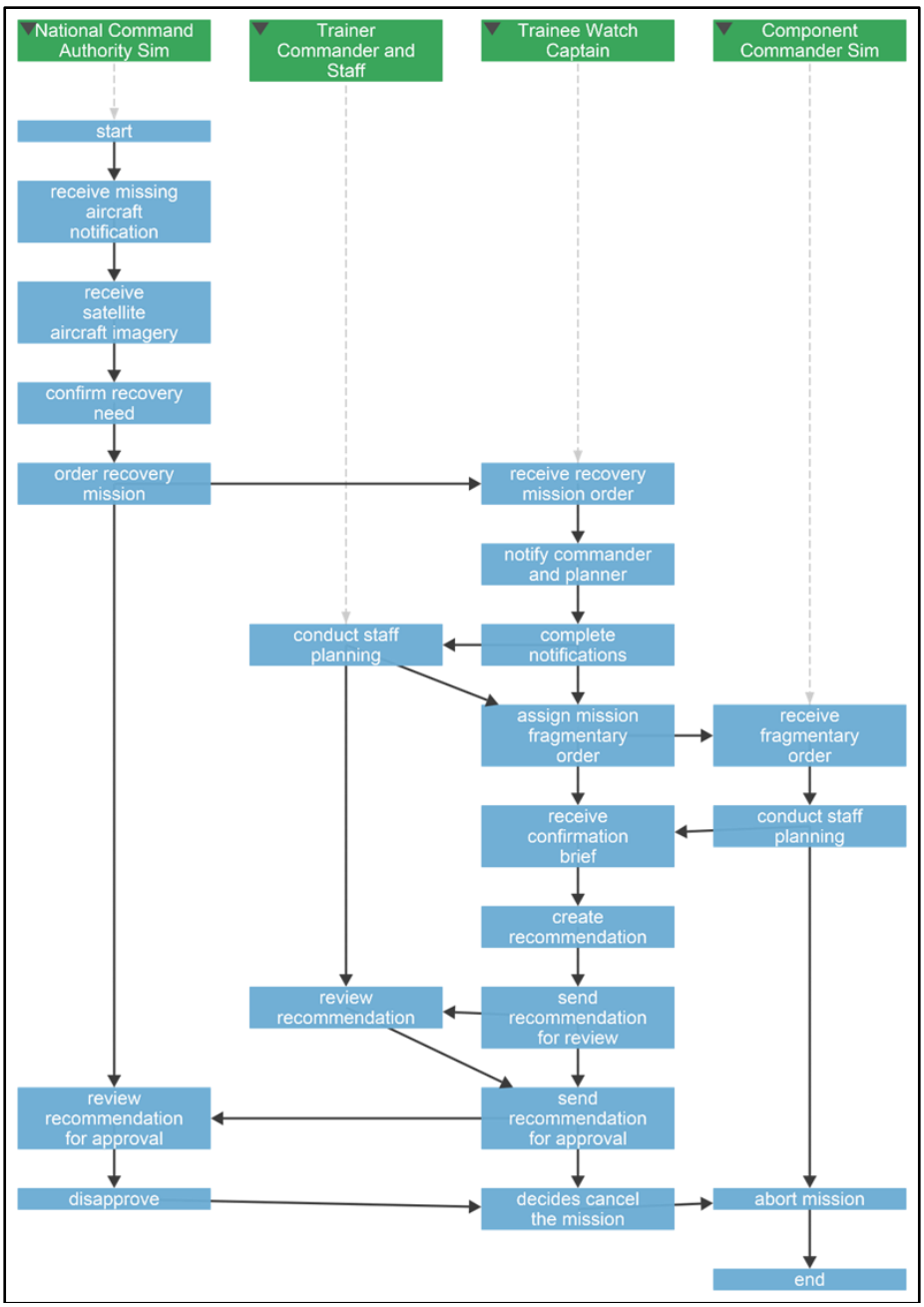

Figure 40. MP sequence diagram for disapproval.

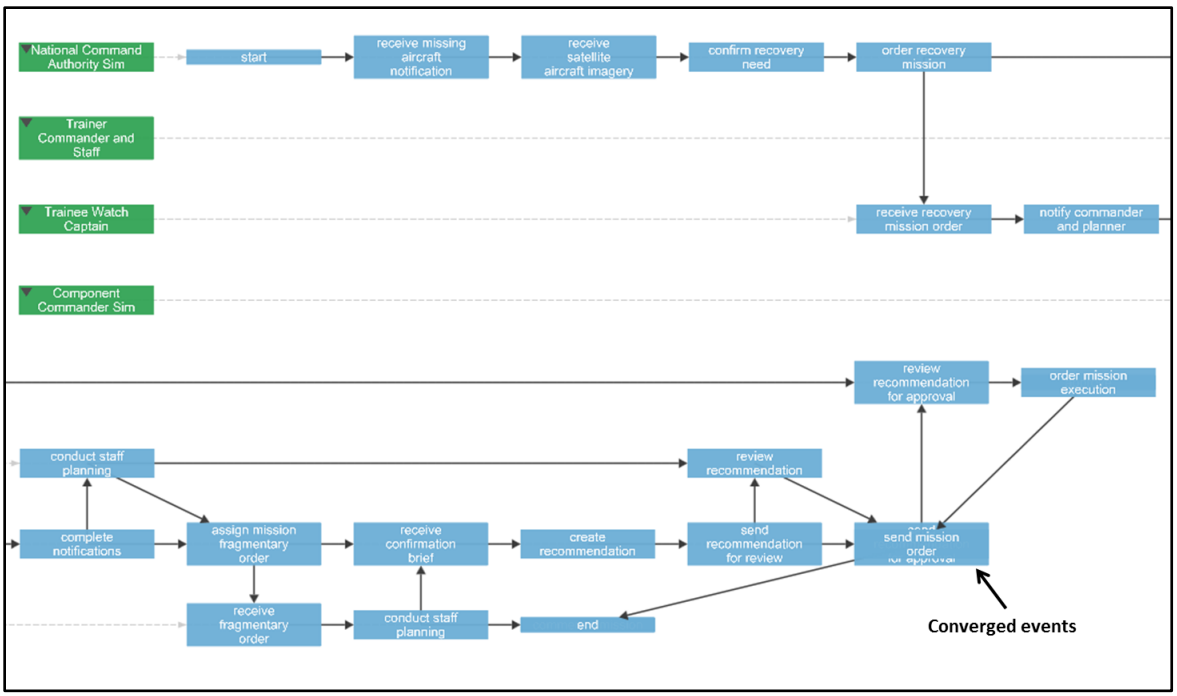

Figure 41. MP swim lane diagram for approval event trace one with converged events (model is split in half for visual representation only).

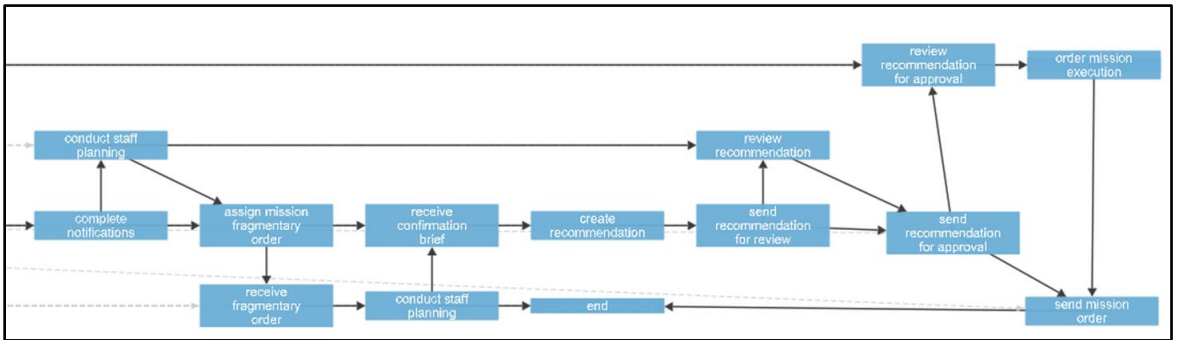

Figure 42. Corrected MP swim lane diagram for approval event trace (right half of model is shown for visual representation only).

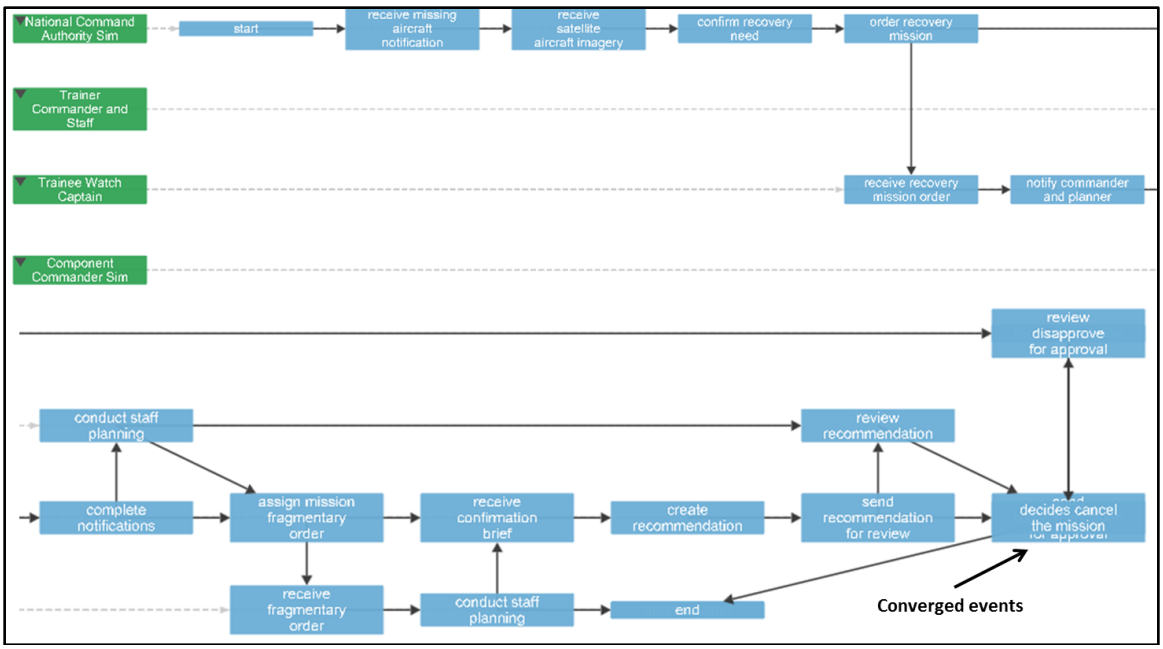

Figure 43. MP swim lane diagram for disapproval event trace one with converged events (model is split in half for visual representation only).

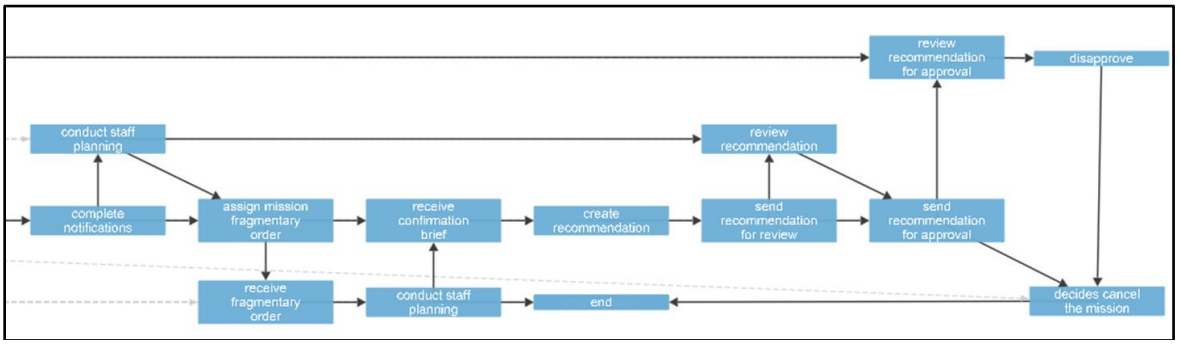

Figure 44. Corrected MP swim lane diagram for disapproval event trace (right half of model shown for visual representation only).

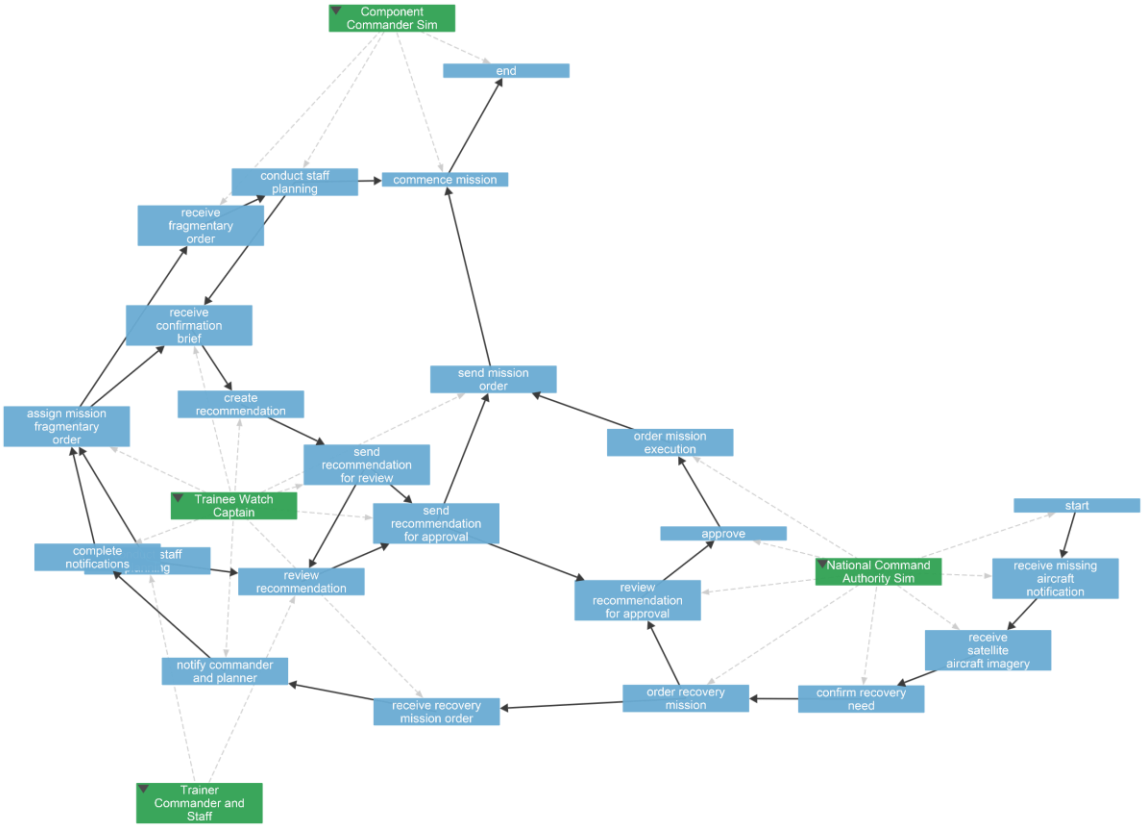

Figure 45. MP force diagram for approval event trace.

d. OV-6c (SvcV-10c and SV-10c) MP Summary

Business process modeling is a valuable tool for identifying, understanding, and transforming the activities and information an organization uses to execute its business or mission. The Object Management Group Business Process Modeling and Notation (BPMN) website states, "BPMN is targeted at a high level for business users and at a lower level for process implementers." The BPMN model is further refined with implementation details by the systems team (OMG 2011). BPMN is one of the methods recommended by the DODAF for development of the event trace description models, which include the OV-6c, SvcV-10c and SV-10c.

Since the core concept for MP is behavior, business process models are a natural fit for the MP approach. Many organizations embark on the business process modeling effort with a burst of zealous energy, transforming any willing and available resource into a business process modeler. As a result, models of all levels of abstraction, complexity,

and logical correctness result from these well-intentioned efforts. As demonstrated with this use case, the system architect will need to update the process model to resolve obvious errors and map the BPMN constructs to the MP approach.

The OV6-c, SvcV-10c, and SV-10c models benefit the architecture by providing a clear advantage in early validation and verification of the model through the generation of the event traces using the MP approach. The baseline model generated only one event trace exposing design errors very early, which were corrected with the revised model. At scope three, MP generates 56 event traces, which provide "immediate, visualized, and exhaustive feedback for model testing" (Auguston et al. 2015). The ability to generate an exhaustive set of scenarios within a given scope is unique to the MP approach; existing BPMN tools cannot guarantee these results according to Giammarco (pers comm.).

7. Services and Systems Functionality Description: SvcV-4/SV-4:

The DODAF SvcV-4 and SV-4 models graphically represent service/system functions and the data flows between them. In MP, functions and data flows are modeled as events. Therefore, these two models can be coded and generated in MP by simply mapping functions and data flows to events. As a use case, the following SvcV-4 is modeled.

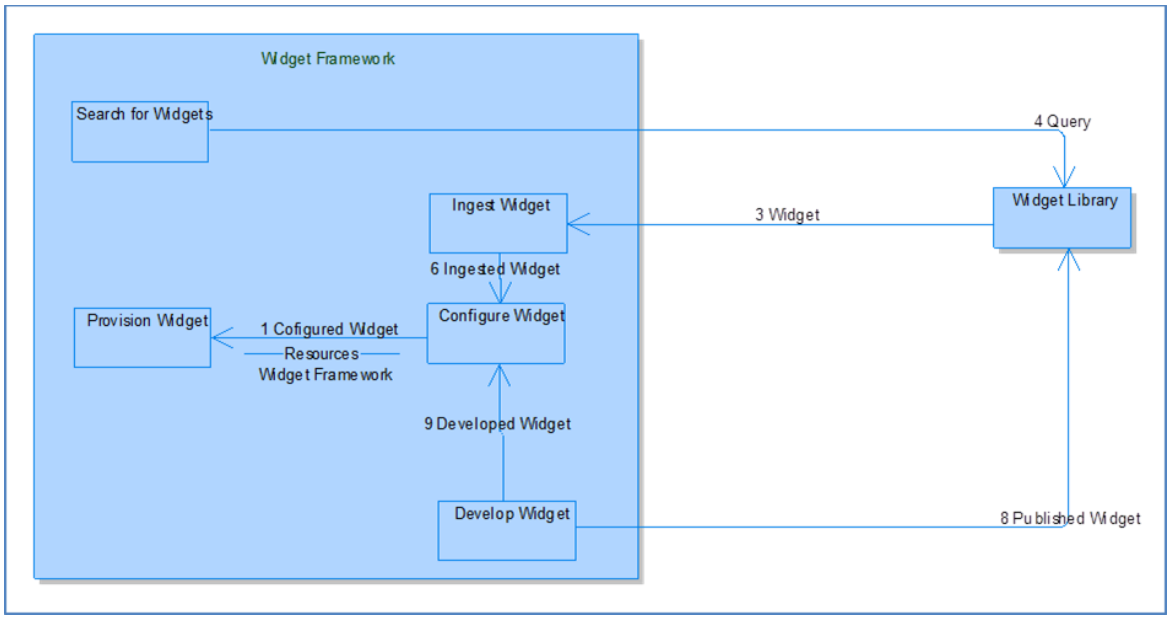

Figure 46. DODAF services functionality description, SvcV-4 (from SPAWAR Pacific 2015).

a. Method of Conversion

Converting the SvcV-4 to MP is simple and straightforward. First, the root events are identified: *Widget_Library* and *Widget_Framework*. Next, the service functions are modeled as events in the roots. Finally, the interactions are coded between the two root events. It is possible to further decompose the events between the functions within the root; however, this is not required to demonstrate the ability to generate the DODAF model using MP.

b. SvcV-4 MP Code

Below is the code-developed model the SvcV-4.

```
SCHEMA SystemView4
/* ------------------------------------------------------
                    Events
------------------------------------------------------ */
ROOT Widget_Library:
               get_search_query
               get_published_widget
               send_widget;
ROOT Widget_Framework:
```

```
                send_search_query
                send_published_widget
                ingest_widget;
/* ------------------------------------------------------
                          INTERACTIONS
------------------------------------------------------ */
COORDINATE
          $x: send_search_query    FROM Widget_Framework,
          $y: get_search_query     FROM Widget_Library
          DO ADD    $x PRECEDES $y; OD;
COORDINATE
          $x: send_published_widget     FROM
          Widget_Framework,
          $y: get_published_widget FROM Widget_Library
          DO ADD    $x PRECEDES $y; OD;
COORDINATE
          $x: send_widget     FROM Widget_Library,
          $y: ingest_widget   FROM Widget_Framework
          DO ADD    $x PRECEDES $y; OD;
```

c. SvcV-4 MP Visualizations

Figure 47 shows the three MP generated visualizations.

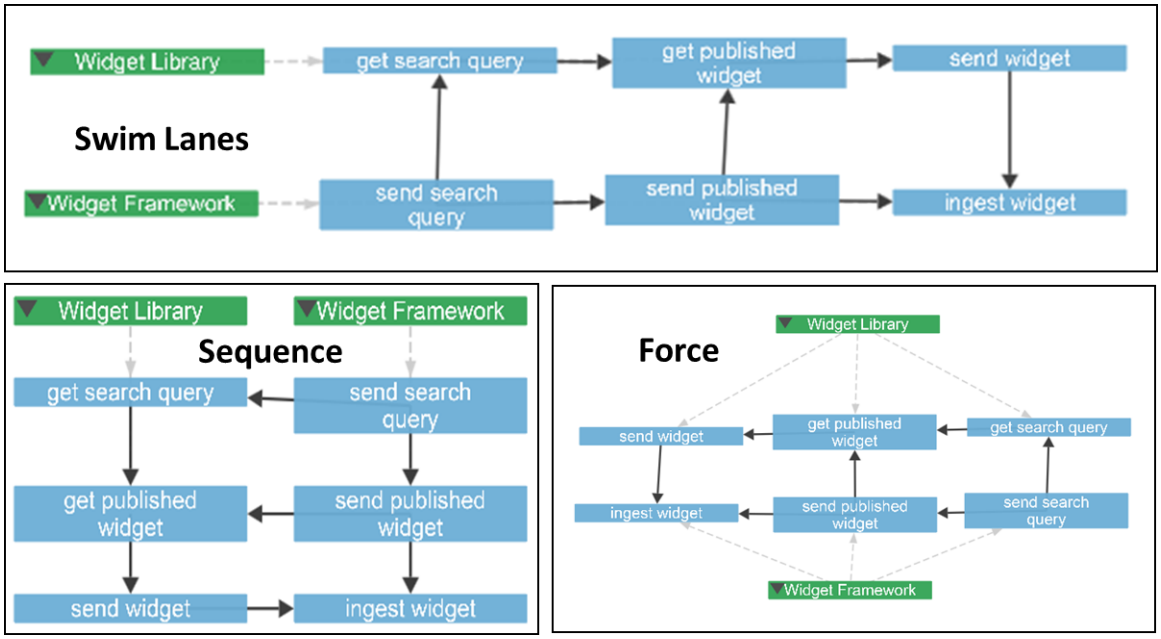

Figure 47. MP swim lanes, sequence, and force visualizations generated from SvcV-4.

d. SvcV-4 and SV-4 Summary

Since functions are synonymous to events in MP, coding SvcV-4 and SV-4 models in MP is a natural fit for realizing MP benefits. The MP swim lane and sequence visualizations are excellent representations of the example SvcV-4 and are easy to read and interpret. As modeled for this research, only one event trace is generated.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

The complexities of today's systems development efforts demand more effective methods and approaches to improve successful outcomes. For more than 50 years, the systems and software development communities have focused on devising methods, approaches, and frameworks to improve the outcomes. These include the waterfall approach, spiral development, rapid prototyping, object oriented methodologies, agile development, scrum techniques, service-oriented architectures, computer-aided software engineering tools, and, more recently, model based systems engineering. Significant improvements are simply not being realized. As discussed, MP introduces a new approach to the mix, and the results of this research reveal much promise for improving systems development through the simple, early discovery of behaviors through the generation of an exhaustive set of use cases (trace events).

This research looks specifically at the ability to use MP-generated models to satisfy DODAF guidelines for compliance. Generating MP models and realizing the early benefits of designing only for intended system behaviors while satisfying the DOD compliance requirements will help socialize the use of the MP approach to DOD program leaders. This research explores three questions:

- What DODAF viewpoints and models can be derived using the MP architecture description approach and language?
- What visualizations could be added to the MP prototype, MP Analyzer, to enhance usage of the MP approach?
- Can DODAF views and models be used to demonstrate the strength of MP to expose high level design errors and unintended system behaviors?

Using the criteria established in this research, the data available in the MP approach generates models from five of the eight DODAF viewpoints, for a total of sixteen of the 51 total DODAF models. The summary of models is shown in Table 10.

Table 11. Summary of DODAF models generated using MP.

Viewpoints	**DODAF Models Generated from MP**
All	None
Capability	CV-2
Data and Information	None
Operational	OV-2, OV-4, OV-5a, OV-5b, OV-6b, OV-6c
Project	PV-1
Services	SvcV-2, SvcV-4, SvcV-10b, SvcV-10c
Standards	None
Systems	SV-2, SV-4, SV-10b, SV-10c

This research includes a mapping of DODAF concepts and a recommended method of conversion for each model. Since MP's strength is behavior modeling, the generated models focus on workflows and system top-level behavior models, such as the SV-2, OV-2, SvcV-2, OV-5b, OV-6c, SvcV-10c and the SV-10c.

While MP is able to generate all of the above models, the current MP Analyzer prototype visualizations are limited. Many other commercial tools present better graphical visualizations with more robust manipulation capabilities. Modelers may consider using alternative tools as the MP Analyzer prototype matures. The following chapter details some recommendations for improved MP visualization capabilities.

Using the MP Analyzer for DODAF model development exposes high-level design errors and unintended behaviors as demonstrated in the generation of the state transition diagrams (OV-6b, SvcV-10b, and SV-10b) and the event trace diagrams (OV-6c, SvcV-10c, and SV-10c). For the order processing state transition model, 36 trace events are generated at scope two and 60 trace events at scope three. The MP Analyzer event traces exposed obvious design errors with the process ending in the *waiting* state

and the *"waiting, item_received, waiting"* states occurring after the *cancelled* state as shown in Figure 48.

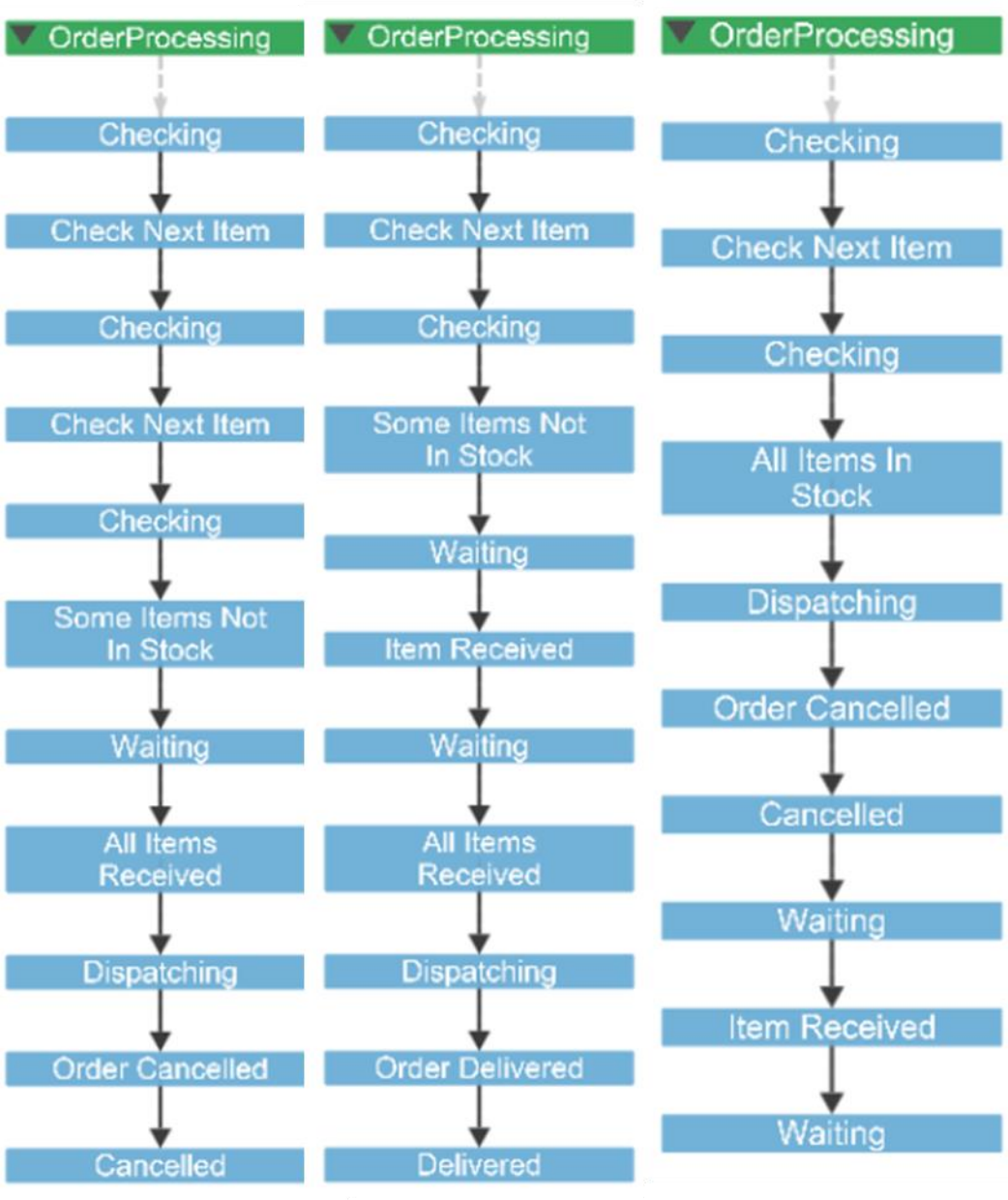

Figure 48. Two valid event trace outcomes (*cancelled, delivered*) and one invalid outcome (*waiting*) discovered using the MP Analyzer.

The baseline event trace diagram (OV-6c) research revealed only one event trace using the MP approach. Analysis conducted from that questionable result identified

design errors resulting in a revised design. The revised event trace diagram (OV-6c), modeled using BPMN, generates 56 trace events at scope three. The system architect now has the ability to analyze and determine if all these behaviors are intended results of the model. No other approach currently provides this ability for early discovery during the initial system-modeling phase.

Finally, the MP Analyzer is an academic tool that is fully open to the systems engineering community and MBSE tool vendors. Researchers hope to inspire systems architects, systems engineers, and industry to adopt the MP approach for its exhaustive scenario generation on a small scope. The MP approach enables the system architect and engineer to focus on reducing design complexity while quickly and easily exposing architectural flaws prior to implementation. The value proposition to DOD programs is the ability to intercept design errors before they become costly system failures or rework requirements. With greater adoption of the MP approach, MBSE vendors can extend the MP capabilities and/or incorporate them in their own tools.

VI. RECOMMENDATIONS

As adoption of the MP approach expands, ease of use will become important consideration in its acceptance. Some areas for consideration include the usability of the prototype tool itself, the ability to integrate the capability into commercial tools, and the training required to develop a cadre of highly skilled MP developers.

As tested for this research, the MP Analyzer tool is a beta prototype. As such, some early limitations exist with the generation of the visualizations. For example, while some manual manipulation of auto-generated diagrams is expected, most diagrams required manual manipulation to uncover stacked events. Although magnification of the visualizations is available, providing more precision in zooming capability would act as a quick improvement. The ability to change the colors of model entities and text is also desirable for mainstream acceptance.

As a prototype, the MP Analyzer tool has a very basic integrated development environment (IDE). Since the MP approach requires development of code, MP tools that provide a robust IDE will aid the MP users in the refinement of their code and models. MP tools should provide some of the following capabilities to make working with MP more efficient for developing and troubleshooting code:

- ability to store, retrieve, and edit MP code within the IDE (this facilitates remote development from any location)
- configuration management
- improved error messaging/handling
- improved code editor (auto language indent for ease of nesting code statements)
- repository of objects names for reuse – this is particularly important to enable the full traceability of the models (from OV-5a to OV-5b, for example)
- ability to describe/define object names
- ability to share models between modelers

A pilot effort should be considered to incorporate the MP approach, language, and algorithms in commercial architecture tools. By doing so, the advantages of the MP language can be realized using the full IDE capabilities of the tool, an integrated architecture database, collaboration and robust visualizations.

The MP approach is still under development and planned extensions are already in development. As such, validation and extension of this research will provide on-going confirmation of the ability to use MP to generate DODAF compliant models. Further research to study and transform complex system architectures that are struggling to meet cost, schedule and performance requirements would be invaluable. Such a study will provide insight on the potential return on investment that can be realized using the MP approach.

APPENDIX A. SIMPLIFIED MP CODE FOR EXAMPLE VISUALIZATIONS

The following code was developed as a simplified model from the revised use case to demonstrate the type of graphical visualizations generated from the MP Analyzer and Eagle6 prototype tools.

```
SCHEMA ResponseMissionTraining
/* ------------------------------------------------------
          PERFORMERS (per BPMN swim lanes)
------------------------------------------------------ */
ROOT National_Command_Authority_Sim:
                    start
                    receive_missing_aircraft_notification
                    order_recovery_mission
                    review_recommendation_for_approval
                    (approve_recommendation                |
                    disapprove_recommendation );
                    approve_recommendation:
                    order_mission_execution;
ROOT Trainer_Commander_and_Staff:
                    conduct_staff_planning
                    review_recommendation;
ROOT Trainee_Watch_Captain:
                    receive_recovery_mission_order
                    complete_notifications
                    assign_mission_fragmentary_order;
ROOT      Component_Commander_Sim:
                    receive_fragmentary_order
                    conduct_staff_planning
                    commence_mission
                    end;
/* ------------------------------------------------------
                    INTERACTIONS
------------------------------------------------------ */
COORDINATE
     $x: order_recovery_mission              FROM
     National_Command_Authority_Sim,
     $y: receive_recovery_mission_order      FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:complete_notifications     FROM
     Trainee_Watch_Captain,
```

```
     $y: conduct_staff_planning    FROM
     Trainer_Commander_and_Staff
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x: conduct_staff_planning              FROM
     Trainer_Commander_and_Staff,
     $y: assign_mission_fragmentary_order    FROM
     Trainee_Watch_Captain
     DO ADD $x PRECEDES $y; OD;
COORDINATE
     $x:assign_mission_fragmentary_order     FROM
     Trainee_Watch_Captain,
     $y: receive_fragmentary_order           FROM
     Component_Commander_Sim
     DO ADD $x PRECEDES $y; OD;
```

APPENDIX B. BASELINE DODAF OV-6C USE CASE

The following BPMN model was used as the initial baseline for demonstrating the conversion of a BPMN model using MP. This model was modified to conduct research on more complex constructs of BPMN such as the parallel and decision gates.

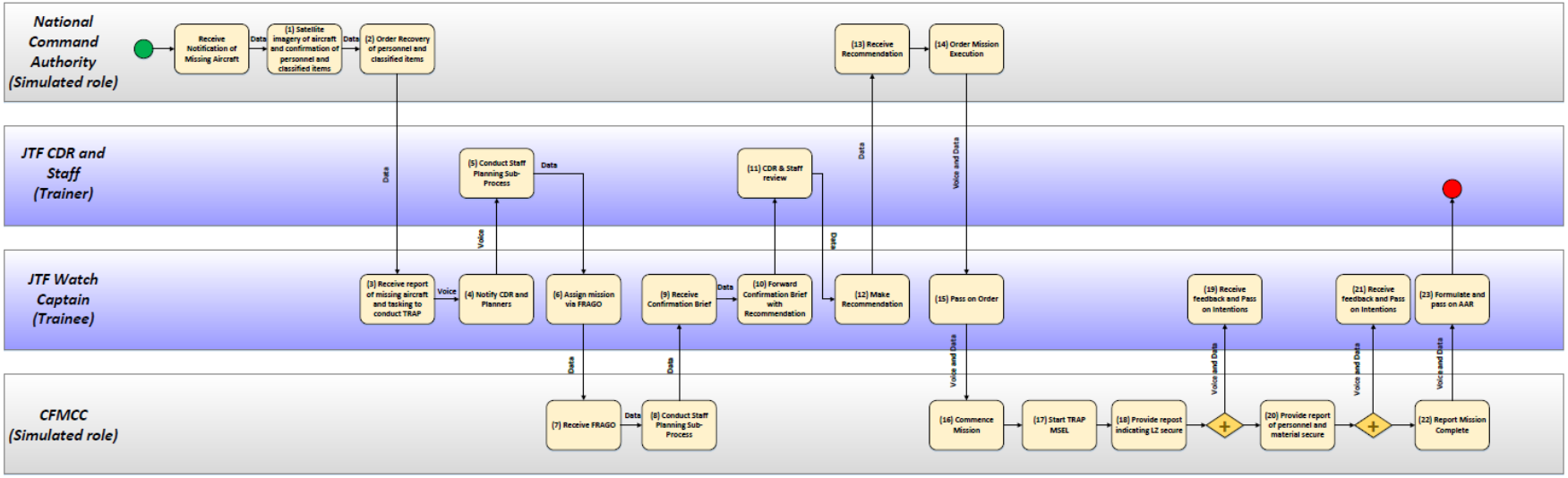

Figure 49. Baseline joint training business process model developed using BPMN (from SPAWAR Pacific 2014b).

A. BASELINE MP CODE USE CASE

The following visualizations and code were generated from the initial BPMN case study model shown in Figure 49.

```
// May 2015
//-------------------------------------------------------
//   Response Mission BPMN Model
//
//   Baseline Model from Initial Research
//   Joanne Pilcher & Kristin Giammarco
//-------------------------------------------------------
ROOT      National_Command_Authority:
          receive_notification_of_missing_aircraft
          satellite_imagery_of_aircraft
          confirmation_of_personnel_and_classified_items
          order_recovery_of_personnel_and_classified_items
          receive_recommendation
          order_mission_execution;
ROOT Joint_Task_Force_Commander_and_Staff:
          conduct_staff_planning_subprocess
          commander_and_staff_review;
ROOT Joint_Task_Force_Watch_Captain:
          receive_report_of_missing_aircraft
          receive_tasking_to_conduct_trap
          notify_cdr_and_planners
          assign_mission_via_FRAGO
          receive_confirmation_brief
          forward_confirmation_brief_with_recommendation
          make_recommendation
          pass_on_order
          receive_feedback_and_pass_on_intentions_A
          receive_feedback_and_pass_on_intentions_B
          formulate_and_pass_on_AAR;
ROOT CFMCC:
          receive_FRAGO
          conduct_staff_planning_sub_process
          commence_mission
          start_TRAP_MSEL
          provide_repost_indicating_LZ_secure
          provide_report_of_personnel_material_secure
          report_mission_complete;
//-------------------------------------------------------
// Simulating COORDINATE with SHARE ALL for Eagle6
//-------------------------------------------------------
//COORDINATE
```

```
//   $x: order_recovery_of_personnel_and_classified_items
//   FROM  National_Command_Authority,
//   $y: receive_report_of_missing_aircraft
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;
ROOT Interaction1:
     (*(order_recovery_of_personnel_and_classified_items
     receive_report_of_missing_aircraft) *);

National_Command_Authority,Interaction1
SHARE ALL order_recovery_of_personnel_and_classified_items;

Joint_Task_Force_Watch_Captain, Interaction1
SHARE ALL      receive_report_of_missing_aircraft;

//COORDINATE
//   $x: notify_cdr_and_planners
//   FROM  Joint_Task_Force_Watch_Captain,
//   $y: conduct_staff_planning_subprocess
//   FROM Joint_Task_Force_Commander_and_Staff
//   DO ADD $x PRECEDES $y OD;
ROOT Interaction2:
     (*(notify_cdr_and_planners
     conduct_staff_planning_subprocess) *);

Joint_Task_Force_Watch_Captain, Interaction2
SHARE ALL      notify_cdr_and_planners;

Joint_Task_Force_Commander_and_Staff, Interaction2
SHARE ALL      conduct_staff_planning_subprocess;

//COORDINATE
//   $x: conduct_staff_planning_subprocess
//   FROM  Joint_Task_Force_Commander_and_Staff,
//   $y: assign_mission_via_FRAGO
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD $x PRECEDES $y OD;
ROOT Interaction3:
     (*(conduct_staff_planning_subprocess
     assign_mission_via_FRAGO) *);

Joint_Task_Force_Commander_and_Staff, Interaction3
SHARE ALL      conduct_staff_planning_subprocess;

Joint_Task_Force_Watch_Captain, Interaction3
SHARE ALL      assign_mission_via_FRAGO;
```

```
//COORDINATE
//   $x: assign_mission_via_FRAGO
//   FROM  Joint_Task_Force_Watch_Captain,
//   $y: receive_FRAGO
//   FROM CFMCC
//   DO ADD    $x PRECEDES $y OD;
ROOT Interaction4:
     (* (assign_mission_via_FRAGO   receive_FRAGO) *);

Joint_Task_Force_Watch_Captain, Interaction4
SHARE ALL      assign_mission_via_FRAGO;

CFMCC, Interaction4
SHARE ALL      receive_FRAGO;

//COORDINATE
//   $x: conduct_staff_planning_subprocess
//   FROM CFMCC,
//   $y: receive_confirmation_brief
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;
ROOT Interaction5:
     (*(conduct_staff_planning_subprocess
     receive_confirmation_brief) *);

CFMCC, Interaction5
SHARE ALL      conduct_staff_planning_subprocess;

Joint_Task_Force_Watch_Captain, Interaction5
SHARE ALL      receive_confirmation_brief;

//COORDINATE
//   $x: forward_confirmation_brief_with_recommendation
//   FROM Joint_Task_Force_Watch_Captain,
//   $y: commander_and_staff_review
//   FROM Joint_Task_Force_Commander_and_Staff
//   DO ADD    $x PRECEDES $y OD;
ROOT Interaction6:
     (*(forward_confirmation_brief_with_recommendation
     commander_and_staff_review) *);

Joint_Task_Force_Watch_Captain, Interaction6
SHARE ALL forward_confirmation_brief_with_recommendation;

Joint_Task_Force_Commander_and_Staff, Interaction6
```

```
SHARE ALL      commander_and_staff_review;

//COORDINATE
//   $x: commander_and_staff_review
//   FROM Joint_Task_Force_Commander_and_Staff,
//   $y: make_recommendation
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;
ROOT Interaction7:
     (* (commander_and_staff_reviewmake_recommendation) *);

Joint_Task_Force_Commander_and_Staff, Interaction7
SHARE ALL      CDR_and_staff_review;

Joint_Task_Force_Watch_Captain, Interaction7
SHARE ALL      make_recommendation;

//COORDINATE
//   $x: make_recommendation
//   FROM Joint_Task_Force_Watch_Captain,
//   $y: receive_recommendation
//   FROM  National_Command_Authority
//   DO ADD    $x PRECEDES $y OD;

ROOT Interaction8:
     (* (make_recommendation   receive_recommendation) *);

Joint_Task_Force_Watch_Captain, Interaction8
SHARE ALL      make_recommendation;

National_Command_Authority, Interaction8
SHARE ALL      receive_recommendation;

//COORDINATE
//   $x: order_mission_execution
//   FROM  National_Command_Authority,
//   $y: pass_on_order
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;
ROOT Interaction9:
     (* (order_mission_execution   pass_on_order) *);

National_Command_Authority, Interaction9
SHARE ALL      order_mission_execution;

Joint_Task_Force_Watch_Captain, Interaction9
```

```
SHARE ALL      pass_on_order;

//COORDINATE
//   $x: pass_on_order
//   FROM Joint_Task_Force_Watch_Captain,
//   $y: commence_mission
//   FROM CFMCC
//   DO ADD    $x PRECEDES $y OD;

ROOT Interaction10:
     (* (pass_on_order   commence_mission) *);

Joint_Task_Force_Watch_Captain, Interaction10
SHARE ALL      pass_on_order;

CFMCC, Interaction10
SHARE ALL commence_mission;

//COORDINATE
//   $x: provide_repost_indicating_LZ_secure
//   FROM CFMCC,
//   $y: receive_feedback_and_pass_on_intentions_A
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;

ROOT Interaction11:
     (*(provide_repost_indicating_LZ_secure
     receive_feedback_and_pass_on_intentions_A)*);

CFMCC, Interaction11
SHARE ALL      provide_repost_indicating_LZ_secure;

Joint_Task_Force_Watch_Captain, Interaction11
SHARE ALL      receive_feedback_and_pass_on_intentions_A;

//COORDINATE
//   $x: provide_report_of_personnel_material_secure
//   FROM CFMCC,
//   $y: receive_feedback_and_pass_on_intentions_B
//   FROM Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;

ROOT Interaction12:
     (*(provide_report_of_personnel_material_secure
     receive_feedback_and_pass_on_intentions_B) *);
```

```
CFMCC, Interaction12
SHARE ALL      provide_report_of_personnel_material_secure;

Joint_Task_Force_Watch_Captain, Interaction12
SHARE ALL      receive_feedback_and_pass_on_intentions_B;

//COORDINATE
//   $x: report_mission_complete
//   FROM CFMCC,
//   $y: formulate_and_pass_on_AAR
//   FROM  Joint_Task_Force_Watch_Captain
//   DO ADD    $x PRECEDES $y OD;

ROOT Interaction13:
     (*(report_mission_complete   formulate_and_pass_on_AAR)
     *);

CFMCC, Interaction13
SHARE ALL      report_mission_complete;

Joint_Task_Force_Watch_Captain, Interaction13
SHARE ALL      formulate_and_pass_on_AAR;
```

B. BASELINE MP VISUALIZATIONS FROM EAGLE6

Eagle6, at the time of this writing, generates horizontal and vertical visualizations as shown below for illustrative purposes. These visualizations are not easy to read and difficult to use as communication tools with stakeholders.

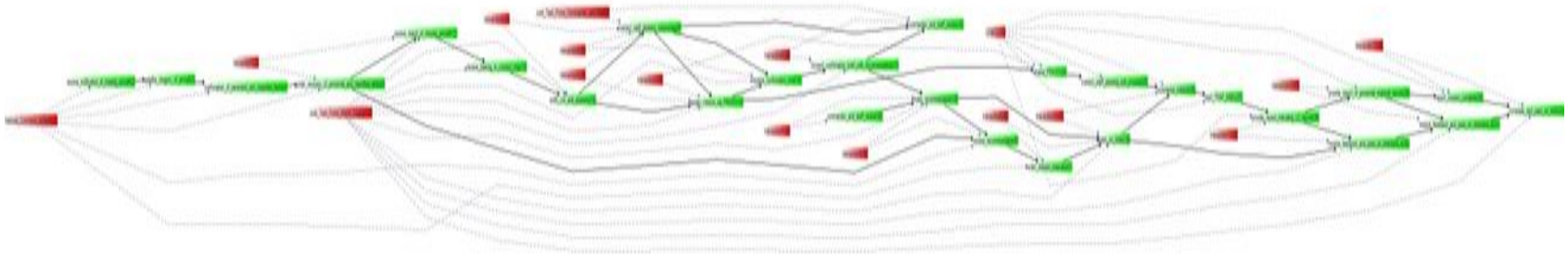

Figure 50. Horizontal Eagle6 visualization.

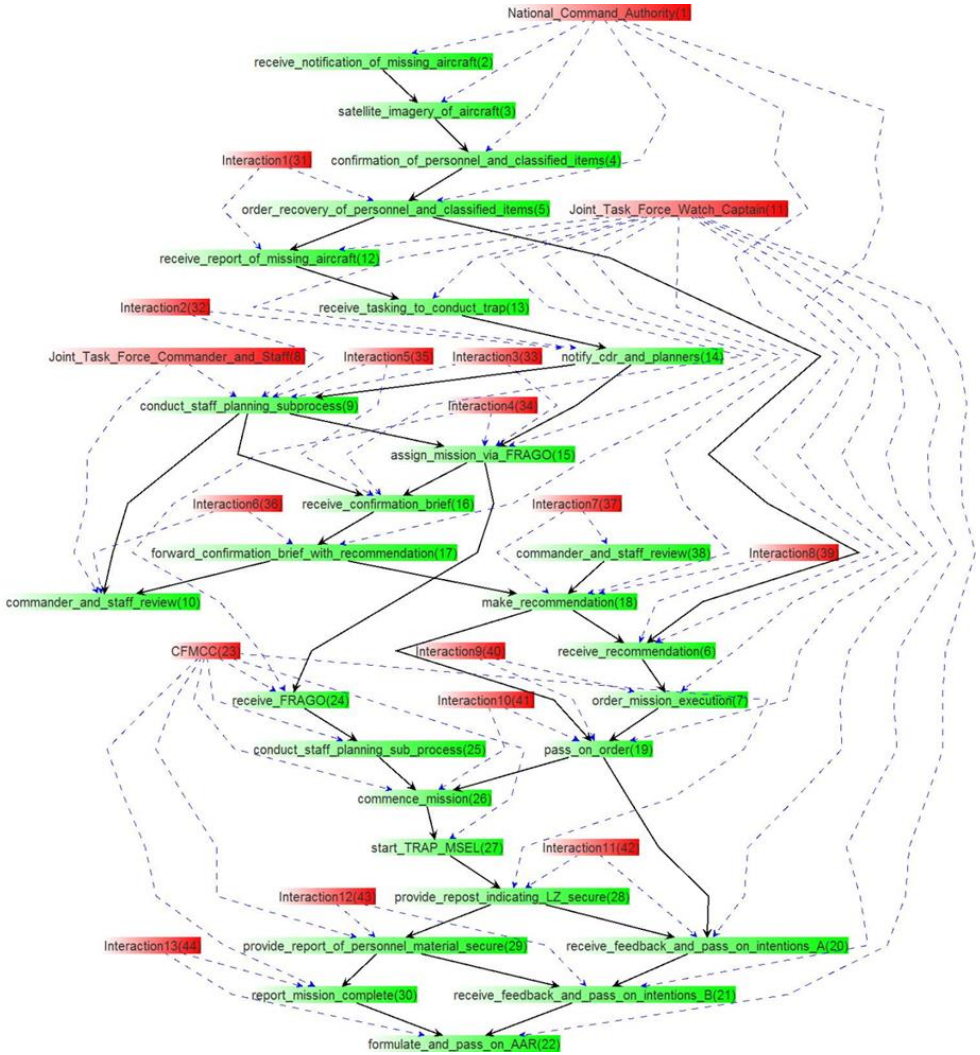

Figure 51. Vertical Eagle6 visualization.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Auguston, Mikhail. 2014. *Behavior Models for Software Architecture*. Monterey, CA: Naval Postgraduate School.

———. 2015. *Monterey Phoenix System and Software Architecture Modeling Language (Version 2.0)*. Monterey, CA: Naval Postgraduate School.

Auguston, Mikhail, Kristin Giammarco, W. Clifton Baldwin, Ji'on Crump, and Monica Farah-Stapleton. 2012. "Controlling Design Complexity with the Monterey Phoenix Approach." Washington, DC, Complex Adaptive Systems, 2012.

Auguston, Mikhail, Kristin Giammarco, W. Clifton Baldwin, Ji'on Crump, and Monica Farah-Stapleton. 2015. "Modeling and Verifying Business Processes with Monterey Phoenix." In Lecture Notes in Procedia Computer Science: Vol 44, 2015 Conference on Systems Engineering Research, 345-353. Waltham, MA: Elsevier B.V.

Blanchard, Benjamin S., and Wolter J. Fabrycky. 2011. *Systems Engineering and Analysis*. Upper Saddle River, NJ: Prentice Hall.

Brooks, Frederick P. 1987. "No Silver Bullet, Essence and Accidents of Software Engineering." *Computer Magazine*, April 1987.

Dam, Steven H. 2014. *DOD Architecture Framework 2.0*. Manassas, VA: SPEC Innovations.

Defense Acquisition University (DAU). 2015. "7.8. the Clinger-Cohen Act (CCA) -- Subtitle III of Title 40 United States Code (U.S.C.)." July 5. https://acc.dau.mil/CommunityBrowser.aspx?id=511635#7.8.2.

Demarco, Tom. 1979. *Structure Analysis and System Specification*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Department of Defense, Deputy Chief Information Officer. 2004. *DOD Architecture Framework Version 1.0*. Washington, D.C.: Department of Defense Chief Information Officer.

———. 2015. *DOD Architecture Framework Version 2.02*. Washington, D.C.: Department of Defense Chief Information Officer.

Department of Defense (DOD). 2015a. *DOD Architecture Framework Version 2.02, Change 1, Volume 1: Overview and Concepts*. Washington, D.C.: Department of Defense Chief Information Officer.

———. 2015b. *DOD Architecture Framework Version 2.02, Change 1, Volume 2: Architecture Data and Models*. Washington, D.C.: Department of Defense Chief Information Officer.

Erl, Thomas. 2009. *SOA Design Patterns*. Boston, MA: SOA Systems.

Erl, Thomas, Anish Karmarkar, Priscilla Walmsley, Hugo Haas, Umit Yalcinalp, Canyang Kevin Liu, David Orchard, Andre Tost, and James Pasley. 2009. *Web Service Contract Design and Versioning for SOA*. Boston, MA: SOA Systems,.

Fowler, Martin, and Kendall Scott. 1997. *UML Distilled*, edited by J. Carter Shanklin. Reading, MA: Addison Wesley Longman.

Giammarco, Kristin, and Auguston, Mikhail. "Monterey Phoenix Concept Mapping." Monterey Phoenix Concept Mapping. Naval Postgraduate School, last modified June 6, 2015, accessed July 2015, https://wiki.nps.edu/display/MP/Concept+Mapping.

———. "Monterey Phoenix Main Event Grammar." Monterey Phoenix Main Event Grammar. Naval Postgraduate School, last modified June 6, 2015, accessed July 2015, https://wiki.nps.edu/display/MP/Event+Grammar.

———. "Monterey Phoenix Main Principles and Advantages." Monterey Phoenix Main Principles and Advantages. Naval Postgraduate School, last modified June 6, 2015, accessed July, 2015, https://wiki.nps.edu/display/MP/Main+Principles+and+Advantages.

Giammarco, Kristin, Monica Farah-Stapleton, and Mikhail Auguston. 2014. *Behavioral Modeling of System Architecture with Monterey Phoenix*. Monterey, CA: Naval Postgraduate School.

INCOSE. 2010. *Systems Engineering Handbook, A Guide for System Life Cycle Processes and Activities*, edited by Cecilia Haskins. San Diego, CA: International Council on Systems Engineering.

Jackson, Daniel. 2012. *Software Abstractions*. Cambridge, MA: The MIT Press.

Langford, Gary O. 2012. *Engineering Systems Integration*. Boca Raton, FL: CRC Press.

Maier, Mark W., and Eberhardt Rechtin. 2009. *The Art of Systems Architecting*. Boca Raton, FL: CRC Press.

NPS. "Monterey Phoenix Home." Monterey Phoenix., last modified June 6, 2015, accessed July, 2015, https://wiki.nps.edu/display/MP/Monterey+Phoenix+Home.

*Object Management Group (*OMG). 2011. *Object Management Group. "Documents Associated with Business Process Model and Notation (BPMN) Version 2.0*: Object Management Group.

Rivera. "Run Eagle6 Beta," accessed July, 2015, http://eagle6modeling.riverainc.com/.

Schwaber, Ken. 2004. *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press.

Sowell, Kathie P. 2006. *The C4ISR Architecture Framework: History, Status, and Plans for Evolution*. McLean, VA: MITRE.

SPAWAR Pacific. "Joint Training Enterprise Architecture: As-Is. Model." San Diego, CA: Space and Naval Warfare Systems Center Pacific.

———. "Joint Training Enterprise Architecture: Systems Engineering Plan." San Diego, CA: Space and Naval Warfare Systems Center Pacific.

———. "Joint Training Enterprise Architecture: To-be, Increment 3." San Diego, CA: Space and Naval Warfare Systems Center Pacific.

———. "Simulation Exercise: Rapid Response Mission Workflow." Business Process Model.

Vaneman, Warren, 2015. "Introduction to Systems Architectures." In Lecture, Naval Postgraduate School, Monterey, CA, October 2.

Yourdon, Edward, and Carl Argila. 1996. *Case Studies in Object Oriented Analysis & Design*. Upper Saddle River, NJ: Prentice-Hall.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California